Vérification formelle et apprentissage logique pour la modélisation qualitative à partir de données single-cell

Samuel Buchet

CENTRALE NANTES

THÈSE DE DOCTORAT DE

L'ÉCOLE CENTRALE DE NANTES

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par
Samuel BUCHET

Vérification formelle et apprentissage logique pour la modélisation qualitative à partir de données *single-cell*

Thèse présentée et soutenue à l'Ecole Centrale de Nantes, le 14 mars 2022
Unité de recherche : UMR 6004, Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

Madalena CHAVES	Directrice de recherche, INRIA Sophia Antipolis
Jean-Paul COMET	Professeur des universités, Ecole Polytechnique de l'Université Côte d'Azur

Composition du Jury :

Président :	Philippe DAGUE	Professeur émérite, Université Paris Saclay
Examinatrice :	Sabine PERES	Professeure des universités, Université de Lyon
Dir. de thèse :	Morgan MAGNIN	Professeur des universités, École Centrale de Nantes
Co-dir. de thèse :	Olivier ROUX	Professeur des universités, École Centrale de Nantes

Remerciements

Bien que le doctorat soit une aventure plutôt personnelle, aller au bout de ces 4 ans n'aurait évidemment pas été possible sans les nombreuses personnes de mon entourage. Entre pandémie, crises mondiales et changements personnels, cette thèse a été réalisée dans une atmosphère particulière et il m'aura fallu de nombreux efforts pour ne pas abandonner. Vient donc le moment de tenter de remercier les personnes qui m'ont accompagné et qui m'ont permis d'aller au bout.

J'aimerais tout d'abord remercier mes directeurs de thèse, Morgan et Olivier, pour m'avoir donné l'opportunité de faire cette thèse, pour leur encadrement ainsi que pour la liberté qu'ils m'ont donné par rapport à mon sujet. Merci également aux membres de mon jury pour leur expertise, et en particulier à Madalena Chaves et Jean-Paul Comet pour avoir accepté d'être rapporteurs. Je voudrais remercier les collègues avec lesquels j'ai eu l'opportunité de travailler, ne serait-ce que pour une discussion autour d'un café. Je remercie Maxime et Tony qui m'ont beaucoup aidé pour les aspects de vérifications formelles et d'inférence logique, ainsi que Xinwei, Honglu et Mitsuhiro pour leurs remarques pertinentes. Je remercie également Inoue-sensei pour m'avoir permis de réaliser mon stage au Japon, ainsi que Florian et l'ensemble des stagiaires et doctorants du Inoue lab au NII à Tokyo pour les discussions intéressantes, les conseils, la bonne humeur et les afterworks. Je remercie chaleureusement Mickaël et Francesco de l'institut Imagine pour leur intérêt dans notre travail, pour m'avoir permis d'utiliser leur données ainsi que pour toute leur aide avec le travail d'analyse qui nous aura mené à notre publication.

Je tiens à remercier l'ensemble des personnels et doctorants que j'ai côtoyé à Centrale, à l'université et en dehors. Merci à Valentin et René pour la bonne ambiance du bureau, Ludi et Julian, Guillaume, Maël, Vanessa, Adrien(s), Rémi, Lolee, Nicolas, Mehdi et bien d'autres... pour les pots et les soirées. Je remercie mes amis qui m'ont entouré pendant la thèse, dont Yoan et Valentin pour leur soutien. Enfin, je tiens à remercier ma famille pour m'avoir toujours soutenu dans mes projets, avec une pensée particulière pour mon père qui me manque désormais beaucoup, et dont la curiosité m'a toujours inspiré.

Table des Matières

INTRODUCTION

Contexte et motivations

La compréhension du fonctionnement des organismes biologiques implique une bonne connaissance de l'activité génétique présente à l'intérieur des cellules. Le comportement des gènes apporte en effet de nombreuses informations sur une multitude de mécanismes tels que la différenciation cellulaire ou encore l'horloge circadienne. Cependant, l'étude des interactions génétiques n'est pas aisée à cause du nombre de gènes impliqués dans le fonctionnement des mécanismes biologiques, et de la difficulté de mesurer leur expression. Bien que le développement des technologies de séquençage des gènes ait grandement amélioré la résolution avec laquelle il est possible d'étudier les systèmes biologiques, les données issues de ces technologies sont généralement bruitées, de grande dimension, et en quantité limitée, ce qui rend leur exploitation complexe. Par ailleurs, l'étude des systèmes biologique bénéficie également des travaux de modélisation des réseaux de régulation génétiques. Ces approches consistent à représenter le système comme un réseau d'interaction entre différents gènes afin de leur associer des rôles fonctionnels. Plus particulièrement, les méthodes développées en informatique, basées sur une modélisation discrète et dynamique du système, ont permis de faciliter l'étude de certains comportements via l'analyse de propriétés dynamiques. Cependant, l'utilisation de ces modèles est complexe, car leur élaboration est chronophage et le niveau d'abstraction qu'ils adoptent par rapport au système réel rend leur validation difficile.

Dans ce travail, nous nous intéressons à la modélisation, à l'aide de l'informatique, des réseaux de régulation à partir de données de séquençage. Un premier aspect important de ce travail est celui de la pluridisciplinarité entre l'informatique et la biologie. Nous adoptons en effet une approche pluridisciplinaire dans le but de traiter au mieux la spécificité de la biologie pouvant rendre difficile la modélisation informatique. Le deuxième aspect qui nous motive particulièrement est celui de la modélisation à partir de données. Le domaine de l'apprentissage automatique, qui repose souvent sur l'utilisation intensive de données, donne effectivement des perspectives intéressantes dans une approche de mo-

délisation automatique. Nous détaillons ces deux éléments de motivation dans la suite de cette partie, au regard des contributions scientifiques qui ont émergé en ce sens.

Informatique pour la biologie

Avant de détailler les différentes approches entre informatique et biologie, nous souhaitons évoquer brièvement certains travaux ayant informellement inspiré notre démarche. La pluridisciplinarité est en effet un aspect qui nous semble intéressant dans un cadre de recherche scientifique par ses apports en termes de motivations ainsi qu'en termes de contribution scientifique. Par exemple, les travaux d'Alan Turing sur la morphogenèse, publiés dans (A. Turing, 1952), illustrent l'application d'une recherche théorique à l'étude des motifs (rayures, spirales, etc.) rencontrés dans le monde du vivant. À l'inverse, on trouve également des exemples de recherche appliquée, notamment dans des contextes industriels, qui ont bénéficié à des domaines plus théoriques. Par exemple, le célèbre test de Student en statistique, publié dans (Student, 1908) et popularisé par Ronald Fisher, a été mis au point par William Sealy Gosset dans le cadre de son travail dans une brasserie.

Plus spécifiquement, l'application de l'informatique à la biologie est intéressant à plusieurs niveaux et cette combinaison peut contribuer à chacune de ces deux disciplines. À l'échelle moléculaire, l'étude des structures de certaines molécules présente un intérêt en informatique car cette question peut être traitée sous l'angle de la combinatoire. Par exemple, des approches basées sur l'optimisation combinatoire ont été proposées pour la prédiction de la structure des molécules d'ARN (Ponty, 2020). Certains travaux proposent également de concevoir des modèles de calcul à partir du comportement de ces mêmes molécules (Qiu et al., 2013), dans le but d'utiliser la biologie comme support de l'informatique. Les interactions entre différentes molécules ont également été étudiées à partir de modèles sous formes de langages, via une description à l'aide de règles d'évolution (Danos et al., 2007). À un niveau d'abstraction plus élevé, la modélisation des réseaux de régulation génétique vise à interpréter les mécanismes de régulation de l'expression génétique d'un système biologique comme un réseau d'interaction global entre différents gènes. C'est sur ce thème que nous nous concentrons dans cette thèse. Le parallèle entre l'informatique et la biologie est intéressant dans ce cadre car les multiples fonctions assurées par les cellules d'un organisme pourraient être assimilées (de manière très informelle) à un programme informatique complexe (Bornholdt, 2008). Les difficultés de ce domaine résident dans le fait que la modélisation des interactions chimiques est complexe et

que les données sur l'expression génétique d'un système sont souvent difficiles à obtenir et très bruitées. De plus, les connaissances de ce domaine sont généralement abstraites et descriptives et peu d'hypothèses de modélisation sont exploitables.

Modélisation informatique à partir de données

La notion de données est importante dans un contexte de recherche entre informatique et biologie puisque ce type de recherche repose généralement sur le traitement de données expérimentales (par exemple avec le traitement de séquences ADN). Nous adoptons dans cette thèse une démarche dans l'esprit du *machine learning* car cette approche facilite la modélisation "automatique" et abstraite, et apporte une dimension intéressante en informatique. La question des données en informatique est en effet abordée par Alan Turing dans (A. M. Turing, 1950), où le principe de l'apprentissage automatique à partir de données émerge du constat que pour la résolution de certains problèmes, l'écriture d'algorithmes peut se révéler particulièrement longue et fastidieuse. Le *machine learning* permet ainsi automatiquement de découvrir des relations entre des variables présentes dans des données, et de spécifier à partir des données des problèmes complexes à définir. Les récents progrès en *machine learning*, notamment basés sur les méthodes de *deep learning* ont donné une grande visibilité au potentiel de modélisation dans ce contexte. Par exemple, certains modèles prédictifs en *deep learning* sont actuellement plus performants que les médecins pour le diagnostic de cancers. En effet, il est montré dans (Kim et al., 2020) que dans le cas du cancer du sein, un modèle de *deep learning* obtient un score de diagnostic $AUROC$ de 0.940 alors que des radiologistes obtiennent un score de 0.810. Le *deep learning* a aussi été utilisé pour la biologie, avec le modèle *alphafold* permettant de prédire la structure de protéines (Jumper et al., 2021). Cependant, l'utilisation des méthodes de machine learning requiert tout de même plusieurs précautions. Leur succès est en effet restreint à certains problèmes bien identifiés comme le traitement d'images ou celui du langage naturel. Les modèles en *machine learning* ont par ailleurs le défaut d'être difficiles à interpréter humainement, là où les modèles formels sont généralement certifiables à l'aide d'outils de *model checking*. La difficulté de l'interprétabilité des modèles automatiques pose notamment des questions éthiques puisque certaines propriétés de sûreté des modèles peuvent être impossibles à établir formellement. Cet aspect est important dans le cas de la modélisation des réseaux génétiques puisque les modèles utilisés n'ont généralement pas directement un but prédictif, mais sont plutôt exploités via des méthodes d'analyse de propriétés dynamiques. Notre démarche vise ainsi à trouver un

compromis entre la possibilité d'exploiter automatiquement des données afin de concevoir des modèles, et la possibilité d'analyser ces modèles dans le but de mieux comprendre les systèmes étudiés.

Problématique

Nous traitons dans cette thèse la problématique de la modélisation discrète et formelle des réseaux de régulation génétique. Bien que de multiples outils tels que les technologies de séquençage permettent d'étudier l'expression génétique dans les cellules de manière très précise, les interactions entre gènes, qui ont une influence sur différents mécanismes biologiques tels que la différenciation cellulaire, ne sont pas toujours bien comprises. Nous avons ainsi pour objectif d'étudier les mécanismes de régulation génétique sous l'angle de la modélisation **qualitative**. Cette approche de modélisation, qui a émergé à la fois en informatique et en biologie, consiste à modéliser les réseaux génétiques à l'aide de modèles type "systèmes de transitions", basés sur des formalismes logiques. Les systèmes de transitions, que l'on rencontre généralement dans le domaine du *model checking* en informatique, modélisent des relations logiques entre différentes variables symboliques et permettent ainsi d'adopter un haut niveau d'abstraction pour la représentation des multiples comportements d'un système. Ce niveau d'abstraction est en effet utile dans un contexte de modélisation génétique car les données disponibles sont généralement bruitées, et les connaissances établies dans ce domaine ont également une forme abstraite.

La question principale que nous abordons est celle de la conception de modèles qualitatifs à partir de données d'expression génétique. Nous nous intéressons en particulier aux données issues des technologies de séquençage *single-cell*, qui apportent une précision inédite dans l'étude des systèmes biologiques. Notre démarche vise ainsi à étudier la mise en place d'une méthode complète de modélisation qualitative, comprenant l'inférence d'un modèle à partir des données *single-cell*, l'analyse des propriétés du modèle inféré, et la validation des comportements du modèle à l'aide d'observations expérimentales et de connaissances biologiques. D'un point de vue informatique, notre intérêt réside dans le développement de nouvelles méthodes d'apprentissage logique adaptées pour la modélisation des systèmes de transitions, et d'algorithmes performants pour l'analyse des modèles. Nous nous intéressons notamment à la formalisation du problème de l'inférence à partir de données, ainsi qu'à la résolution des problèmes algorithmiques rencontrés. Du point

de vue biologique, nous voyons par ailleurs un double objectif. Premièrement, le développement d'outils de modélisation doit permettre de mieux comprendre les mécanismes d'interactions génétiques à l'œuvre au sein des cellules d'un organisme. Deuxièmement, l'application de la modélisation sur les données de séquençage *single-cell* a également pour objectif de proposer de nouveaux outils d'analyse permettant d'aller plus loin dans l'exploitation de ces données.

Contributions

Nos contributions s'organisent autour de trois parties différentes. Dans une première partie, nous traitons le problème de l'analyse des modèles qualitatifs de réseaux génétiques. Nous proposons en particulier une nouvelle méthode pour le problème de l'accessibilité dans les réseaux d'automates. La deuxième partie concerne les algorithmes d'inférence de modèles qualitatifs. Dans celle-ci, nous développons un nouvel algorithme d'inférence logique adapté aux données bruitées. Enfin, la troisième partie traite de l'application de ce nouvel algorithme sur un exemple concret avec des données collectées à partir des technologies de séquençage *single-cell*. Nous détaillons dans la suite les contributions développées dans chacune de ces trois parties.

Résolution de l'accessibilité dans les réseaux d'automates

Les modèles qualitatifs sont intéressants pour représenter des systèmes avec un grand nombre de variables, mais cela rend leur analyse plus difficile du point de vue de la complexité algorithmique. Nous proposons donc une nouvelle méthode de résolution pour le problème d'accessibilité dans les réseaux d'automates asynchrones, dans le but de montrer que des approches de résolution efficaces peuvent être mises en place pour ce type de problème. Notre méthode combine la technique du *Bounded Model Checking* avec une approche basée sur l'analyse statique. L'application du *Bounded Model Checking* sur les réseaux d'automates a été mise en place via le développement d'un encodage de la dynamique des réseaux d'automates sous forme de problème de satisfaisabilité booléenne (SAT). Concernant l'analyse statique, nous proposons une extension à partir des travaux réalisés dans (Folschette et al., 2015b ; Paulevé, 2018 ; Paulevé et al., 2012). Cette extension résulte en la mise en place d'une nouvelle propriété permettant de surestimer rapidement la longueur des solutions pour une instance de problème d'accessibilité donnée, afin de garantir la complétude de l'approche en *Bounded Model Che-*

cking. Ces travaux ont fait l'objet d'une implémentation disponible en ligne à l'adresse https://github.com/smbct/AAN-reach.git.

Inférence de programmes logiques sur données bruitées

Les données d'expression génétiques étant généralement bruitées, cela complique l'inférence de modèles abstraits tels que les modèles qualitatifs. Pour répondre à ces difficultés, nous développons un nouvel algorithme d'inférence de programmes logiques basé sur un framework de modélisation existant nommé *LFIT* (Ribeiro et al., 2018b). Pour adapter *LFIT* au contexte des données expérimentales bruitées, nous proposons des hypotheses de modélisation qui diffèrent de celles utilisées dans les algorithmes existants. À partir de ces hypothèses, nous proposons une nouvelle méthode d'inférence basée sur des notions d'optimisation combinatoire. Ce nouvel algorithme, nommé *Learning Optimized Logical Hypothesis* (*LOLH*), a été publié dans (Buchet et al., 2021) et a également fait l'objet d'une implémentation en ligne disponible à l'adresse https://github.com/smbct/LOLH.git.

Application de l'inférence sur données *single-cell*

Nous appliquons notre méthode d'inférence *LOLH* sur un jeu de données *single-cell* afin de montrer ses bénéfices, et découvrir des relations génétiques dans ces données. Une des difficultés principales en modélisation qualitative réside par ailleurs dans la validation des modèles inférés. Nous illustrons donc une manière de combiner l'algorithme *LOLH* à des méthodes d'analyse de graphes afin de déterminer si le modèle obtenu modélise le système réel de manière pertinente. Nous proposons une première application de notre algorithme, consistant à mettre en évidence de manière non supervisée les gènes co-exprimés et les différents types des cellules présentes dans les données. Cette application a été publiée dans (Buchet et al., 2021). Nous proposons également une seconde application permettant d'interpréter les données *single-cell* comme des séries temporelles, dans le but d'inférer des relations dynamiques entre les gènes.

Organisation du manuscrit

Ce manuscrit s'organise en 4 chapitres. Dans le chapitre 1, nous présentons le contexte de la modélisation dynamique des systèmes biologiques. Nous présentons en particulier le domaine de la modélisation qualitative, ainsi que les différents formalismes proposés en ce

sens. Dans ce chapitre, nous introduisons enfin les principales questions traitées dans ce contexte, qui sont celles de l'analyse et de l'inférence des modèles qualitatifs. Le chapitre 2 est consacré au problème de l'analyse des modèles qualitatifs. Nous abordons de manière générale la difficulté algorithmique des problèmes traités et les différentes approches algorithmiques proposées dans ce cadre. Nous nous concentrons ensuite sur le problème d'accessibilité des réseaux d'automates asynchrones. Nous proposons une nouvelle méthode de résolution, basée sur une combinaison entre une approche de *Bounded Model Checking* et une approche reposant sur l'analyse statique. Les chapitres 3 et 4 sont enfin dédiés au problème de l'inférence de modèles qualitatifs. Dans le chapitre 3, nous introduisons ce problème plus en détail, et nous résumons plusieurs approches existantes pour le traiter. Nous présentons le framework de modélisation *LFIT*, et nous en proposons une extension avec la proposition de notre nouvel algorithme *LOLH* pour traiter les données expérimentales. Le chapitre 4 est finalement consacré à l'application de notre algorithme d'inférence sur des données obtenues avec des technologies de séquençage *single-cell*. Dans ce chapitre, nous présentons les technologies *single-cell* ainsi que les méthodes de calcul développées pour analyser ces données. Nous appliquons enfin notre algorithme d'inférence sur un jeu de données particulier afin d'exploiter à la fois des relations de co-expression et des relations dynamiques entre les gènes à partir de ces données.

Collaborations

Cette thèse a donné lieu à deux collaborations. Une première collaboration a permis la réalisation d'un séjour de recherche de 6 mois à partir de mars 2019, au sein du Inoue laboratory, au National Institute of Informatics [1] (NII) à Tokyo. Ce séjour, supporté financièrement par le NII, entre dans le cadre du MOU avec le RFI Atlanstic [2] 2020, et fait également suite à une collaboration de plusieurs années entre le Inoue laboratory [3] et l'équipe de recherche MéForBio du LS2N. Ce travail encadré par le professeur Katsumi Inoue a permis une prise de connaissances approfondie des travaux réalisés autour du framework *LFIT*, initialement développé au NII. La recherche effectuée à cette occasion a été à l'origine de notre nouvel algorithme d'inférence pour *LFIT*, présenté dans le chapitre 3.

1. https://www.nii.ac.jp/en/
2. https://atlanstic2020.fr/
3. http://research.nii.ac.jp/il/index21.html

Une deuxième collaboration a été réalisée à partir de septembre 2020 avec l'équipe de recherche de Mickaël Ménager à l'institut Imagine [4]. Cette collaboration à distance a permis l'obtention du jeu de données *single-cell* utilisé dans cette thèse. Le travail effectué à cette occasion a consisté à appliquer notre algorithme d'inférence dans un contexte de recherche concret en biologie. Cette collaboration nous a notamment permis de profiter de l'expertise des chercheurs de l'institut Imagine afin d'interpréter correctement les résultats obtenus avec notre méthode, ce qui a débouché sur nos travaux publiés dans (Buchet et al., 2021).

Notations

Ensembles et multiensembles Un ensemble d'éléments A_1 comprenant les élément a et b est noté $\{a, b\}$, avec unicité des éléments. La cardinalité de A_1, c'est-à-dire son nombre d'éléments, est notée $|A_1|$, avec $|A_1| = |\{a, b\}| = 2$ dans cet exemple. L'ensemble vide est noté $\varnothing$, avec $|\varnothing| = 0$. Un multiensemble (avec répétitions) A_2, comprenant deux fois un élément a, trois fois un élément b et une fois un élément c, est noté $\{\!\{a, a, b, b, b, c\}\!\}$. La cardinalité de A_2 est notée $|A_2|$, avec $|A_2| = |\{\!\{a, a, b, b, b, c\}\!\}| = 6$ ici.

Produit cartésien Le produit cartésien de deux ensembles est noté $\times$, avec $\{a, b\} \times \{c, d\} = \{(a, c), (a, d), (b, c), (b, d)\}$. On a également $\{a, b\}^2 = \{a, b\} \times \{a, b\}$. Enfin, l'opérateur $\prod$ est également utilisé pour le produit cartésien d'ensembles indicés.

Intervalles entiers Les intervalles sur les nombres entiers sont notés $[\![m, n]\!]$, où $[\![m, n]\!] = \{m, m + 1, .., n - 1, n\}$, avec $m, n \in \mathbb{N}$ et $m \leq n$. De plus, $|[\![m, n]\!]| = n - m + 1$.

Intervalles réels Les intervalles sur les nombres réels sont notés $[c, d]$, avec $c, d \in \mathbb{R}$ et $c \leq d$.

Séquences Une séquence d'éléments x comprenant $k \in \mathbb{N}$ éléments est notée $(x^1 :: x^2 :: .. :: x^{k-1} :: x^k)$, où x^i désigne le i-ème élément de la séquence. La longueur de x est notée $|x|$, avec $|x| = k$ ici. La séquence vide est enfin notée ε, avec $|\varepsilon| = 0$.

4. https://www.institutimagine.org/en/mickael-menager-187

Atomes logiques Les atomes logiques, représentant les valeurs discrètes de variables logiques multivaluées, sont notés v_i, où v désigne le nom de la variable et i (≥ 0) désigne sa valeur discrète pour l'atome. Dans le chapitre 4, nous employons également la notation $v_{i/j}$, où j désigne la valeur maximale pour la variable v (c'est-à-dire $v \in [\![0, j]\!]$).

Fonction indicatrice La fonction indicatrice pour un ensemble A, appliquée à un élément a et notée $\mathbb{1}_A(a)$ est la fonction retournant 1 si l'élément a donné en argument appartient à A, et 0 sinon. De manière plus formelle, $\mathbb{1}_A(a) = \begin{cases} 1 \ si \ a \in A \\ 0 \ sinon \end{cases}$.

Chapitre 1

Modélisation qualitative en biologie des systèmes

Résumé

Dans ce chapitre, nous introduisons le contexte biologique dans lequel s'inscrit cette thèse. Nous décrivons brièvement les concepts biologiques sur lesquels s'appuie la modélisation des réseaux de régulation, ainsi que les enjeux de cette modélisation pour la recherche. Nous passons en revue les différents formalismes de modélisation proposés pour l'étude des systèmes biologiques, et nous présentons plus en détail la modélisation qualitative, qui sera étudiée dans cette thèse. Nous introduisons enfin différentes notions de ces modèles qui seront réutilisées par la suite et nous donnons les objectifs généraux de ce travail.

1.1 Contexte biologique

L'étude des systèmes biologiques, aussi bien à l'échelle de l'organisme qu'à l'échelle moléculaire, constitue un enjeu de recherche important. La compréhension du fonctionnement des organismes vivants a notamment un impact sur le développement de traitements contre les maladies ou l'étude des effets environnementaux sur la santé. Cependant, la compréhension des mécanismes qui régissent le fonctionnement des organismes biologiques est difficile en raison de leur complexité, du coût et de la difficulté de la réalisation d'expériences, et de la présence d'enjeux éthiques importants. Pour ces raisons, la modélisation des mécanismes biologiques a une grande importance dans la recherche biologique et la recherche biomédicale puisqu'elle vise à en simplifier l'étude, en réduisant les besoins

expérimentaux. Dans ce contexte, la biologie des systèmes est une thématique visant à adopter une vision systémique de la modélisation du vivant (Kitano, 2002). Cette vision est particulièrement utile à l'échelle moléculaire, où les interactions entre gènes produisent des comportements complexes et engendrent des mécanismes qui ne sont pas toujours bien compris, et où les technologies expérimentales permettent par ailleurs d'acquérir de gros volumes de données.

Cependant, l'étude des systèmes biologiques se révèle difficile à cause du bruit des données expérimentales, de la difficulté de leur obtention, ainsi que de l'hétérogénéité des informations observables, ce qui complique la mise en place d'hypothèses de modélisation physique. L'informatique présente alors un double intérêt pour contribuer à la compréhension de ces systèmes. Dans un premier temps, le développement d'outils pour le traitement automatique de données biologiques (en particulier de données génomiques) permet de faire face à l'augmentation du volume de données disponible. Dans un second temps, l'informatique présente un intérêt théorique via le développement de modèles abstraits visant à mieux intégrer les différentes données hétérogènes disponibles. Dans cette thèse, nous nous intéressons en particulier à la modélisation des réseaux génétiques, c'est-à-dire aux interactions possibles entre les gènes d'un organisme, dans le but de comprendre le fonctionnement des mécanismes biologiques qui y sont associés. Dans la partie suivante, nous détaillons brièvement les mécanismes moléculaires sous-jacents à l'expression des gènes dans les cellules. Nous passons ensuite en revue les différentes méthodes de modélisation existantes pour la compréhension de ces phénomènes.

1.1.1 Régulation de l'expression génétique

La cellule, que l'on peut considérer comme la brique fondamentale des organismes biologiques, est un composant complexe délimité par une membrane, et au sein duquel sont présentes de nombreuses molécules continuellement en interaction. On s'intéresse en particulier à l'aspect génétique d'un organisme. En effet, les gènes, correspondant à des morceaux de l'ADN, lui-même présent dans le noyau des cellules, ont un rôle majeur dans le fonctionnement de l'organisme. Ceux-ci sont tous formés à partir des quatre mêmes nucléotides, mais sont pourtant à l'origine de mécanismes variés, ce qui nous encourage à les considérer et à les traiter essentiellement comme de l'information.

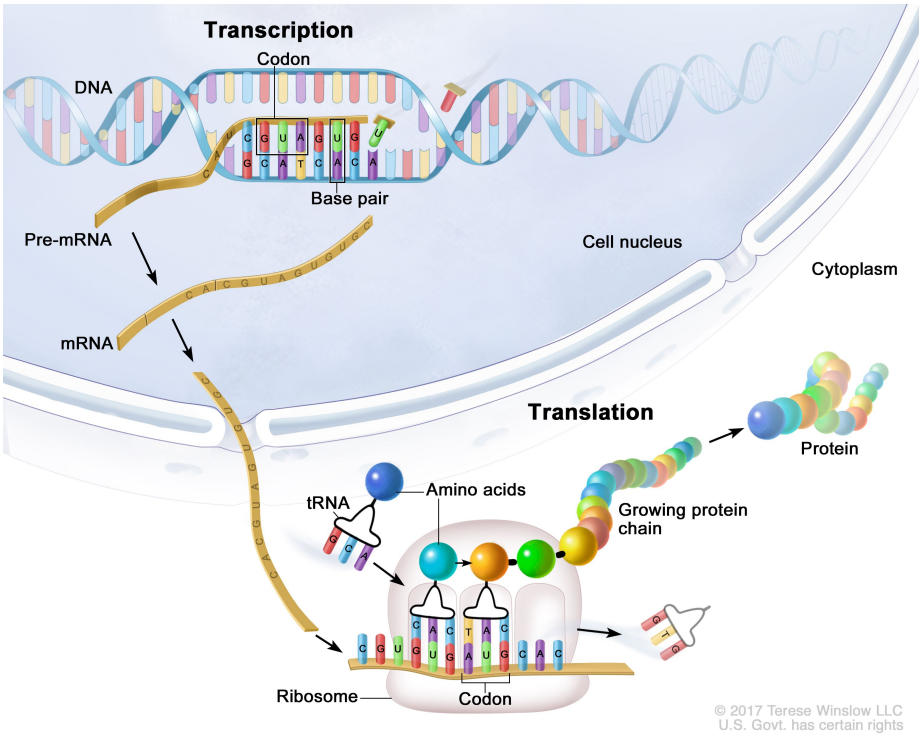

FIGURE 1.1 – Représentation schématique de la transcription des gènes en ARN (à l'intérieur du noyau) et de leur traduction en protéine (à l'extérieur du noyau) au sein d'une cellule. Illustration par Terese Winslow LLC (https://www.teresewinslow.com/).

Le développement de la génomique a suscité beaucoup d'enthousiasme, notamment avec des projets comme le Human Genome Project [1], qui vise à séquencer l'entièreté du génome humain. Cependant, la seule information des gènes ne suffit pas à expliquer toute la complexité d'un organisme. Par exemple, l'hypothèse anciennement soutenue, selon laquelle chaque gène déterminait une unique protéine n'est pas valide étant donné le faible nombre de gènes (~20000 dans les cas du génome humain) (MORAES et al., 2016). Bien que les gènes soient présents de manière presque identique dans les noyaux de toutes les cellules d'un organisme, le mécanisme d'expression d'un gène est un processus dynamique, et ceux-ci ne sont pas exprimés de la même manière dans toutes les cellules. Les gènes subissent en effet plusieurs transformations qui mènent à la synthèse de protéines. Dans un premier temps, le processus de transcription d'un gène permet la création d'une molécule d'ARN messager qui peut se déplacer hors du noyau. Dans un second temps, le processus de traduction du gène permet la synthèse d'une protéine à partir de la lecture de l'information présente sur l'ARN messager par le ribosome. Les protéines ainsi synthétisées

1. https ://www.genome.gov/

peuvent alors avoir plusieurs fonctions à l'intérieur d'une cellule. La figure 1.1 illustre les mécanismes moléculaires de transcription et de traduction des gènes.

Comprendre les mécanismes qui font évoluer l'expression des gènes au sein des cellules est un défi majeur. La connaissance des processus de transcription et de traduction de gènes n'explique pas totalement l'évolution de leur expression. En effet, des processus de régulation sont à l'œuvre afin que l'expression génétique se maintienne à un niveau cohérent, et évolue de façon à assurer le bon fonctionnement de l'organisme (Jacob et al., 1961). Par exemple, certaines protéines appelées facteurs de transcription (*TF*) ont la possibilité de reconnaître certaines séquences ADN et peuvent avoir une influence positive (activation) ou négative (inhibition) sur la transcription des gènes correspondants. Cette vision dynamique de l'expression génétique a notamment été imagée par Conrad H. Waddington dans le cadre de la biologie développementale [2] (Allen, 2015). Il est ainsi intéressant d'interpréter l'ensemble des gènes comme un réseau d'interactions. Bien que la compréhension des mécanismes de régulation constitue une question théorique en elle-même (Bich et al., 2016), le décryptage des interactions génétiques représente un enjeu pour la bonne compréhension des mécanismes biologiques. Nous nous concentrons donc dans ce travail sur la notion de réseau de régulation biologique, et nous tentons de l'utiliser pour expliquer différents phénomènes dynamiques observables.

1.1.2 Données expérimentales sur l'activité génétique

Plusieurs technologies expérimentales ont été développées pour collecter des données sur l'activité moléculaire au sein des cellules d'un organisme. Ces données permettent d'améliorer la compréhension des mécanismes génétiques à l'œuvre au sein des cellules. Elles représentent notamment un intérêt du point de vue de la modélisation de la régulation génétique. Par exemple, les données d'expression génétique témoignent de l'activité des molécules d'ARN messager à l'intérieur des cellules, et constituent un indicateur important sur l'activité d'un système (Delgado et al., 2019). Différentes technologies permettent d'obtenir de telles données. Par exemple, les puces à ADN (*DNA-microarray*) permettent d'obtenir l'expression de dizaines de milliers de gènes. Le séquençage ARN, consistant à lire chaque nucléotide sur les molécules d'ARN, est également très utilisé pour mesurer l'activité des gènes notamment grâce aux technologies de séquençage haut-

2. L'évolution de l'expression génétique de cellules y est apparentée aux trajectoires empruntées par des billes roulant le long d'une pente.

débit. Récemment, le séquençage ARN cellule par cellule (*single-cell RNA sequencing*, que nous dénommerons séquençage *single-cell* par la suite) a par ailleurs été rendu possible, donnant lieu à des données dotées d'une résolution sans précédent, permettant d'analyser l'expression de milliers de gènes dans plusieurs milliers de cellules. Ces technologies sont particulièrement intéressantes pour la modélisation automatique de la régulation génétique et elles seront abordées plus en détail dans le chapitre 4. Enfin, il est également possible de mesurer les interactions entre protéines, par exemple avec la technique de double hybride (*Two-hybrid screening*), ainsi que les interactions entre protéines et gènes.

Dans le cadre de l'étude des mécanismes de régulation génétique, les données expérimentales présentant un aspect temporel sont particulièrement intéressantes, bien que difficiles à obtenir. Les données les plus simples à obtenir dans ce cas sont les données sur les états stables du système, c'est-à-dire les états où le système n'évolue pas. Pour ces dernières, il est par exemple possible de se baser sur des données de perturbation, où un gène est perturbé par rapport à un état de repos, et où le système est mesuré une fois stabilisé. Ce type de donnée est notamment proposé (de manière simulée) dans le challenge d'inférence de modèle *DREAM 4* (Marbach et al., 2010). Les données de séries temporelles, consistant à effectuer des mesures consécutives d'un système à plusieurs instants donnés, semblent les plus intéressantes du point de vue de l'étude de la dynamique car elles apportent une meilleure résolution temporelle sur le système étudié. Cependant, obtenir ce type de données est particulièrement coûteux, et les séries temporelles disponibles possèdent généralement peu de points temporels. Par exemple, la série temporelle utilisée dans (Thorne, 2018) ne comporte que 7 points temporels, et le challenge *DREAM 11* (The Respiratory Viral DREAM Challenge Consortium et al., 2018) met à disposition des séries temporelles comportant une dizaine de points non réguliers. Une alternative aux séries temporelles est par ailleurs apparue récemment avec les technologies *single-cell*. Les méthodes d'inférence de séries pseudo-temporelles permettent en effet d'approximer des séries temporelles avec plusieurs milliers de points. Cette nouvelle source de données prometteuse sera exploitée dans le chapitre 4.

1.1.3 Formalisation des connaissances et modélisation

Devant la complexité des interactions moléculaires qui ont lieu au sein des cellules, la modélisation, notamment sous forme de réseau, est un outil indispensable pour aider

à la compréhension des différents mécanismes biologiques. Une première approche pour rendre les connaissances accumulées exploitables consiste en l'élaboration de cartes d'interactions biochimiques, appelées *biological pathway*. Ces cartes concernent plusieurs types de processus comme le métabolisme, les voies de signalisation à l'intérieur d'une cellule, ou encore les voies de régulation des gènes, décrivant l'évolution de leur expression. Ces connaissances sont généralement compilées dans des bases de données comme celle de KEGG : Kyoto Encyclopedia of Genes and Genome [3]. Elles sont utilisées par les biologistes entre autres pour comprendre les données génomiques obtenues expérimentalement et les mécanismes associés par exemple à des maladies (García-Campos et al., 2015).

Ces cartes d'interactions ne permettent cependant pas de comprendre efficacement certains comportements génétiques à l'intérieur des cellules. Des approches de modélisation ont donc été développées pour décrire le fonctionnement du système à un niveau d'abstraction plus élevé, à partir d'un ensemble d'interactions (the Mutation Consequences and Pathway Analysis working group of the International Cancer Genome Consortium, 2015). Ainsi, les modèles de réseaux de régulation biologiques ont pour but d'expliquer des comportements dynamiques complexes en abstrayant les différentes molécules impliquées, notamment directement sous la forme de gène. Plusieurs types de modèles ont été proposés, par exemple avec l'utilisation de simples graphes, des modèles statistiques tels que les réseaux bayésiens et les réseaux de neurones récurrents (Blasi et al., 2005), des modèles physiques à base d'équations différentielles, ou encore des modèles dits qualitatifs, basés sur des formalismes logiques (de Jong, 2002). Ces différentes approches de modélisation n'ont pas exactement les mêmes objectifs et elles se complètent. Par exemple, les modèles basés sur des équations différentielles proposent une vision quantitative du système, avec une caractérisation précise de l'activité de différentes molécules. Ces modèles sont donc précis mais ils sont en contrepartie coûteux d'un point de vue calculatoire, et ils nécessitent beaucoup de données pour être correctement paramétrés. Au contraire, les modèles qualitatifs adoptent un haut niveau d'abstraction en représentant l'activité de différents composants d'un système sous forme discrète (par exemple en modélisant les gènes de manière booléenne), et peuvent ainsi prendre en compte un nombre très important de composants.

3. https://www.genome.jp/kegg/

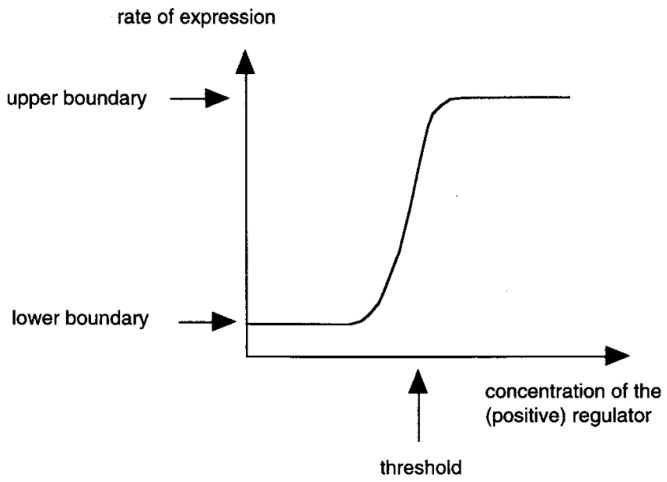

FIGURE 1.2 – Évolution sigmoïdale du taux d'expression d'une molécule en fonction de la concentration d'un régulateur. Schéma tiré de (THOMAS et al., 1995).

Dans cette thèse, nous nous intéressons spécifiquement à la modélisation qualitative. Cette approche a émergé à partir des travaux de René Thomas sur les réseaux multivalués (THOMAS, 1973), ainsi que ceux de Stuart Kauffman sur les réseaux booléens (KAUFFMAN, 1969). La modélisation qualitative vise à simplifier drastiquement la modélisation de chaque composant afin de se concentrer sur une description fidèle du comportement global du système, en représentant les interactions entre ses composants par des relations logiques. Plusieurs éléments encouragent ce type de modélisation. Tout d'abord, les données expérimentales sur les systèmes biologiques sont bruitées et peu précises, en particulier avec les technologies de séquençage ADN. Par ailleurs, une interprétation symbolique de l'évolution de l'activité d'un gène semble adéquate car le taux d'expression d'une molécule évolue généralement de manière sigmoïdale, comme représenté dans la figure 1.2, et la transcription des gènes se produit de manière non régulière dans le temps (LONGO et al., 2006). Ce type de modélisation se révèle enfin particulièrement intéressant d'un point de vue informatique par sa proximité avec certains formalismes de modélisation.

1.2 Modélisation qualitative des réseaux biologiques

Les travaux initiés en parallèle par René Thomas sur les réseaux de Thomas (modèles qualitatifs multivalués) et par Stuart A. Kauffman sur les réseaux booléens ont donné lieu à une recherche active sur la modélisation qualitative/discrète des systèmes biolo-

giques, notamment des réseaux de régulation génétique. La modélisation qualitative se concentre ainsi sur la représentation abstraite du comportement dynamique d'un système biologique. Les composants d'un réseau sont modélisés de manière discrète, et leur évolution est modélisée par des règles symboliques qui représentent les interactions avec les autres composants du réseau. La notion dynamique ne correspond donc pas forcément à un temps, mais plutôt à une succession d'évènements qui font évoluer le système entre plusieurs états. On peut noter par ailleurs que ce type de modélisation est proche de plusieurs autres approches en bio-informatique, dont le langage κ (Danos et al., 2007) qui vise à représenter directement les interactions moléculaires à partir de règles d'association. La modélisation qualitative se distingue de ces approches par rapport au niveau d'abstraction adopté, nous nous concentrons donc sur celle-ci dans cette thèse.

Le niveau d'abstraction de ces modèles est particulièrement utile lorsque le système étudié est difficile à observer, et lorsque les données disponibles sur ce système sont hétérogènes. Les variables discrètes peuvent alors représenter plusieurs concepts différents, par exemple le fait qu'un gène est activé ou non, le dépassement d'un seuil de température, le dépassement d'un seuil de concentration d'une protéine, etc. D'un point de vue génétique, la modélisation discrète semble pertinente car les données acquises avec les technologies de séquençage conduisent facilement à ce type d'interprétation, et c'est une abstraction également utilisée par les chercheurs biologistes. La modélisation qualitative se révèle également intéressante d'un point de vue informatique. En effet, les formalismes développés en modélisation qualitative ont une grande proximité avec ceux développés pour la vérification formelle de programmes, ou encore en logique. La recherche sur les modèles qualitatifs a ainsi bénéficié des connaissances avancées sur l'étude de ces modèles en informatique, menant à des méthodes d'analyse efficaces ainsi qu'à une compréhension théorique avancée de ces différents formalismes. Dans la suite de cette partie, nous passons en revue plusieurs formalismes mis en place pour la modélisation qualitative, et nous développons plusieurs concepts concernant leur étude qui seront utilisés dans les chapitres suivants.

1.2.1 Formalismes de modélisation qualitative

À partir des réseaux de Thomas introduits par René Thomas, et des réseaux booléens proposés par Stuart A. Kauffman, différents formalismes ont émergé pour la modélisation qualitative des réseaux biologiques. Les réseaux booléens modélisent chaque composant

d'un réseau (gène ou protéine par exemple) par une **variable booléenne**, indiquant si le composant est présent/absent (ou encore actif/inactif). L'évolution de chaque composant du réseau est alors déterminée par une fonction booléenne qui dépend généralement de plusieurs autres composants (Bornholdt, 2008 ; Wang et al., 2012). Ces modèles ont été appliqués avec succès notamment pour modéliser le développement de la Drosophile (Gonzalez et al., 2008), ou encore le cycle cellulaire de la levure de bière (Bornholdt, 2008), qui sont des organismes très étudiés en biologie. Les réseaux booléens ont également été proposés sous forme de modèles aléatoires dans (Kauffman et al., 2003) afin d'identifier les relations booléennes d'un réseau de transcription d'une levure, à partir d'un graphe d'influence déjà déterminé. L'approche ensembliste a également été proposée afin de déterminer et caractériser les multiples modèles potentiellement compatibles avec certaines propriétés (Schwab et al., 2020).

Le formalisme des réseaux de Thomas, proposé par René Thomas, se distingue des réseaux booléens par l'utilisation de variables discrètes **multivaluées** plutôt que de variables booléennes (Thomas et al., 1995). Les interactions entre composants sont alors modélisées par des changements de niveau discret conditionnés par le niveau d'activité des autres composants (par exemple, un composant `a` passe du niveau 1 au niveau 2 si le composant `b` est au niveau 0 et le composant `c` est au niveau 1). Cette approche de modélisation a par exemple été utilisée dans (Mateus et al., 2007) pour l'étude du bactériophage lambda. L'outil GINsim [4] implémente également ce type de modèle pour proposer plusieurs analyses (Chaouiya et al., 2012). Le formalisme des Frappes de Processus, introduit dans (Paulevé et al., 2011) et inspiré par l'approche du π-calcul stochastique, peut aussi être considéré comme une formalisation supplémentaire des réseaux de Thomas. Enfin, les réseaux d'automates, qui peuvent être vus comme une extension des Frappes de Processus, ont été proposés dans (Folschette et al., 2015b) comme un cadre formel minimal et intuitif pour la modélisation multivaluée des réseaux biologiques. Dans ce formalisme, les composants biologiques (par exemple les gènes) sont représentés par des automates dont les différents états locaux modélisent leurs niveaux d'activité. Les interactions entre composants y sont modélisées à partir de transitions locales dans les automates, conditionnées par des ensembles d'états locaux sur d'autres automates. En comparaison aux réseaux booléens, les réseaux d'automates se révèlent intéressants car ils permettent de naturellement modéliser les variables multivaluées. De plus, l'utilisation de

4. http://ginsim.org/

transitions locales au lieu de fonctions booléennes permet de modéliser plus directement le non-déterminisme (les réseaux booléens synchrones sont en effet déterministes par nature). Ce type de représentation a entre autres été utilisé dans le but de construire la plus grande dynamique d'un système, afin de procéder ensuite par raffinements successifs pour limiter l'ensemble des comportements possibles. Les réseaux d'automates sont traités dans le chapitre 2 de cette thèse, dans le cadre de l'**analyse** de réseaux.

Autres formalismes réutilisés pour les modèles qualitatifs

La modélisation qualitative des réseaux biologiques a aussi bénéficié des formalismes développés dans d'autres domaines de recherche. Par exemple, les réseaux de *Petri*, très utilisés pour les méthodes formelles, ont été appliqués à la modélisation de réseaux de régulation dans (Comet et al., 2005). Le langage logique de l'*Answer Set Programming* (*ASP*) a aussi été utilisé pour la modélisation qualitative, par exemple dans (Fayruzov et al., 2011). Enfin, la programmation logique inductive a aussi été utilisée dans (Inoue et al., 2014) pour le problème de l'inférence de modèles à partir de données qualitatives dynamiques. Ce formalisme est très similaire aux réseaux d'automates, avec la proximité entre les règles logiques et les transitions locales d'automates. Il s'agit du formalisme qui est adopté aux chapitres 3 et 4, dans le cadre de l'**inférence** de modèles qualitatifs. Enfin, bien que les modèles qualitatifs ont pour but de représenter des comportements plus riches en comparaison à de simples graphes d'interactions, la notion de graphe reste toutefois utilisée dans ce cadre. En effet, plusieurs définitions ont été proposées pour abstraire un modèle qualitatif sous la forme d'un graphe d'influence. Cette notion a été utilisée dans (Comet et al., 2013) afin de déterminer des propriétés théoriques sur les réseaux booléens à partir de la structure de leur graphe d'influence. Cela permet également de réduire la complexité de l'analyse d'un modèle en séparant les composants du réseau indépendants entre eux.

L'exemple suivant définit un modèle jouet de réseau booléen pour un système à 3 composants, ainsi que le graphe d'influence qui y est associé. Cet exemple sera réutilisé pour illustrer les différentes notions concernant les modèles qualitatifs présentées dans la suite de cette partie.

Exemple 1.1. *Soit un réseau booléen de trois variables, défini à partir des trois fonctions de mise à jour suivantes ($x \in \{0,1\}^3$ étant un vecteur booléen déterminant l'état du réseau) :*

- $f_1(x) = x_1 \wedge (x_2 \vee x_3)$
- $f_2(x) = \neg x_2 \wedge x_3$
- $f_3(x) = (x_1 \wedge x_2) \vee x_3$

*Les variables étant booléennes, l'ensemble des états du réseau booléen est $\mathcal{S} = \{0,1\}^3$. Le graphe d'influence de ce réseau booléen, construit selon la définition dans (*Chevalier *et al., 2019), est représenté sur la figure 1.3. Ce graphe contient un unique arc inhibiteur, sur le composant 2. Cet arc se justifie par le fait que pour les paires d'états $(001, 011)$ et $(101, 111)$, pour lesquelles le composant 2 évolue de 0 à 1, la valeur de $f_2(x)$ évolue de 1 à 0.*

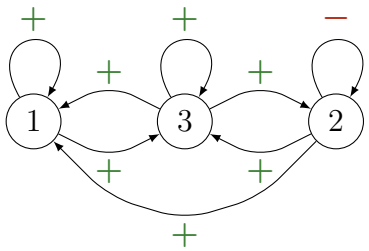

Figure 1.3 – Graphe d'influence du réseau booléen de l'exemple 1.1.

Équivalences et traductions entre modèles

Bien que différents formalismes de modélisation aient été proposés dans la littérature, des équivalences peuvent exister entre certains formalismes et des traductions sont possibles. Par exemple, il a été montré dans (Paulevé, 2018) que les réseaux d'automates peuvent être traduits en réseaux de *Petri* bornés, et vice versa. La traduction du formalisme des Frappes de Processus vers celui des réseaux de Thomas est proposée dans (Folschette et al., 2015a). Enfin, une traduction immédiate des frappes de processus vers le formalisme du π-calcul stochastique a été mise en évidence dans (Paulevé et al., 2011). L'élaboration de traductions entre modèles a plusieurs intérêts. Cela permet par exemple de connaître facilement la complexité des problèmes d'analyse associés aux modèles, ainsi que de profiter des méthodes d'analyse déjà développées pour les formalismes équivalents. En cas de non-équivalence, l'étude des traductions entre modèles permet aussi

de souligner les enrichissements proposés par de nouveaux formalismes. Il est également important de souligner que le haut niveau d'abstraction de la modélisation qualitative complique la comparaison et la reproduction des analyses de systèmes. La recherche a bénéficié d'efforts importants pour faciliter l'accès aux outils existants, leur interconnexion, et la reproduction des résultats publiés, notamment via des outils comme le notebook CoLoMoTo [5] (Naldi et al., 2018).

Enrichissement de la modélisation

Dans le but d'élaborer des modèles qui soient le plus conformes possible à la réalité biologique, plusieurs extensions ont été proposées pour la modélisation qualitative. La notion de temps a en particulier reçu une grande attention dans la modélisation des réseaux biologiques. Certaines extensions visent donc à attribuer un délai particulier pour la transition d'un composant entre deux niveaux discrets. La notion de délai a par exemple été proposée pour développer une extension à partir des réseaux de Thomas dans (Comet et al., 2010). Elle a également été proposée pour le formalisme des Frappes de Processus (Sheikh et al., 2019) ainsi que pour celui des réseaux d'automates (Ben Abdallah et al., 2017a). Cette notion a aussi été utilisée dans le cadre des réseaux de Petri, via leur extension temporisée (L. Chen et al., 2007). L'aspect probabiliste a également été proposé pour les modèles qualitatifs. Par exemple, l'utilisation du formalisme du π-calcul stochastique dans (Paulevé et al., 2011) permet de représenter des délais de transitions aléatoires. Dans le cadre des réseaux booléens, la notion probabiliste a été modélisée en proposant plusieurs fonctions de mise à jour pour chaque composant, avec différentes probabilités (Shmulevich et al., 2002). Enfin, des formalismes hybrides ont également été proposés à partir des réseaux de Thomas dans (Behaegel et al., 2016), ainsi que pour les réseaux booléens dans (Stoll et al., 2012), afin de conserver le niveau d'abstraction des modèles qualitatifs tout en adoptant une représentation continue du temps.

1.2.2 Sémantique et graphe de transitions

Les modèles qualitatifs étant principalement définis à partir des interactions locales entre leurs composants, le comportement global d'un modèle correspond alors à toutes les combinaisons possibles des interactions locales. Nous introduisons dans la suite la notion de sémantique, ou encore mode de mise à jour d'un modèle, qui permet de définir la

5. https://colomoto.github.io/

manière dont se combinent les évolutions locales des modèles. Ce dernier élément permet ensuite de spécifier l'ensemble des comportements du modèle, que l'on peut représenter par un graphe appelé graphe de transitions ici. Nous détaillons également par la suite certaines caractéristiques étudiées sur les graphes de transitions, permettant de caractériser le fonctionnement des modèles.

Sémantique des modèles qualitatifs

La définition d'un modèle de réseau biologique dans un formalisme donné ne permet généralement pas d'en énumérer directement tous les comportements. Ce choix a une grande importance en termes de modélisation puisqu'il influe sur les propriétés d'un système modélisé. La sémantique a également un impact sur l'analyse des modèles qualitatifs car la difficulté des problèmes algorithmiques associés peut en dépendre.

En effet, bien que les interactions entre composants soient explicitement définies dans tout formalisme de modélisation qualitative, la sémantique du modèle (appelée aussi mode de mise à jour), c'est-à-dire le choix des composants à mettre à jour à partir d'un état donné, doit être spécifiée afin de complètement déterminer le comportement de ce modèle. Cette question a été traitée dès l'apparition des premiers modèles qualitatifs, et plusieurs modes de mise à jour ont été proposés dans la littérature. La **sémantique asynchrone**, aussi appelée asynchrone pleine (Paulevé, 2020), initialement introduite par René Thomas, établit qu'un seul composant du réseau peut être mis à jour à la fois dans une transition du modèle. Ainsi, l'état successeur d'une transition dans la sémantique asynchrone ne sera différent que d'un seul composant. Cela permet de considérer le processus d'évolution du modèle séparément de son état (Thomas, 1981). Une des justifications aussi fournies pour cette sémantique est le fait qu'il est infiniment improbable que plusieurs composants évoluent d'un niveau discret à un autre exactement au même moment. Cette hypothèse implique par ailleurs le fait que les composants du modèle ne sont pas forcés à être mis à jour. En effet, cette sémantique autorise un composant donné à rester fixe tandis qu'une multitude d'autres composants peuvent être mis à jour. Enfin, ce mode de mise à jour conduit généralement à des modèles ayant des comportements dynamiques très riches, ce qui facilite la modélisation d'une sur-approximation du système biologique étudié. Un grand nombre de comportements dynamiques peut par ailleurs compliquer l'analyse.

La **sémantique synchrone**, initialement proposée par Stuart Kauffman, établit que tous les composants du modèle doivent être mis à jour simultanément. Dans le cas des réseaux booléens, ce mode de mise à jour conduit à des comportements déterministes pour les modèles : à partir d'un état, le modèle peut évoluer dans exactement un unique autre état (Kauffman, 1969) (on peut noter que ça n'est pas le cas pour des modèles à base de transitions locales comme les réseaux d'automates). Dans cette sémantique, la mise à jour des composants est forcée à chaque évolution du système modélisé. Le déterminisme de cette sémantique conduit à une dynamique plus restreinte que dans le cas asynchrone. Cela peut ajouter des difficultés à la modélisation puisque les modèles synchrones représentent des comportements moins complexes. Néanmoins, cette sémantique peut faciliter l'analyse des modèles comportant un grand nombre de variables. Par ailleurs, nous montrons dans le chapitre 3 que cette sémantique peut avoir un intérêt pour l'apprentissage automatique de modèles qualitatifs.

Ces travaux fondateurs ont également conduit à l'émergence de plusieurs sémantiques alternatives. La **sémantique asynchrone généralisée** par exemple autorise la mise à jour de n'importe quel sous-ensemble de composants du modèle à partir d'un état donné (Paulevé, 2020). D'un point de vue théorique, cette sémantique est plus générale que les sémantiques synchrone et asynchrone, c'est-à-dire que les transitions possibles dans les sémantiques asynchrone et synchrone le sont aussi en sémantique généralisée. On peut également mentionner la sémantique ***Most Permissive*** pour les réseaux booléens, qui a récemment été proposée dans (Paulevé et al., 2020). Cette dernière émerge à partir du constat que certains comportements intuitifs de systèmes biologiques ne peuvent pas être représentés avec la sémantique asynchrone généralisée. La sémantique *Most Permissive* introduit donc des comportements supplémentaires en considérant des états intermédiaires, de manière à augmenter l'expressivité des modèles. Elle possède aussi un intérêt en termes d'analyse car il a été montré que les problèmes d'analyse de modèles ont une complexité algorithmique plus faible dans cette sémantique par rapport aux sémantiques présentés ci-dessus.

Une manière de formaliser la sémantique d'un modèle consiste à énumérer les transitions possibles dans ce modèle. Pour un modèle possédant un ensemble $\mathcal{S}$ d'états discrets, il est donc possible de définir plusieurs sémantiques comme des sous-ensembles $T \subset \mathcal{S} \times \mathcal{S}$ de l'ensemble total des transitions possibles. L'exemple suivant illustre cette définition

avec les sémantiques synchrone, asynchrone et asynchrone généralisée dans le cas du réseau booléen de l'exemple 1.1. Dans cet exemple, les boucles (transition d'un état vers lui-même) ne sont pas autorisées.

Exemple 1.2. *En reprenant l'exemple 1.1, on peut définir l'ensemble des transitions du modèle en fonction d'une sémantique donnée. On représente $T \subset \mathcal{S} \times \mathcal{S}$ l'ensemble des transitions du modèle par un ensemble de couples d'états. Les ensembles de transitions engendrés par les sémantiques synchrone, asynchrone et asynchrone généralisée sont donnés par :*

$$
\begin{aligned}
&- T_{sync} &&= \{(s, s') \in \mathcal{S} \times \mathcal{S} \mid s \neq s', s' = (f_1(s), f_2(s), f_3(s))\} \\
&- T_{async} &&= \{(s, s') \in \mathcal{S} \times \mathcal{S} \mid s \neq s', \exists! i \in [\![1, 3]\!] : s'(i) = f_i(s) \wedge \forall j \in [\![1, 3]\!] \setminus \{i\}, s_j = s'_j\} \\
&- T_{gen} &&= \{(s, s') \in \mathcal{S} \times \mathcal{S} \mid s \neq s', \forall i \in [\![1, 3]\!] : s'_i = s_i \vee s'_i = f_i(s)\}
\end{aligned}
$$

Dans le cas de l'exemple 1.1 (figure 1.4) cela donne :
$T_{sync} = \{(100, 000), (010, 000), (110, 101), (101, 111), (111, 101), (001, 011), (011, 001)\}$.
$T_{async} = \{(110, 111), (111, 101), (101, 111), (110, 100), (100, 000), (010, 000), (001, 011), (011, 001)\}$
$T_{gen} = T_{async} \cup \{(110, 101)\}$

Graphe de transitions des modèles qualitatifs

Lorsque les comportements d'un modèle qualitatif sont complètement spécifiés grâce à sa sémantique, il peut être intéressant d'étudier sa dynamique en la représentant sous forme d'un graphe, que l'on appelle ici **graphe de transitions** (parfois appelé *state transition graph*, ou encore diagramme état-transition). Le graphe de transitions est le graphe orienté dont les sommets sont les états discrets (généralement énumérables) d'un modèle qualitatif, et dont les arcs correspondent aux transitions possibles entre les états dans ce modèle, selon la sémantique adoptée. Généralement, le nombre de sommets d'un tel graphe est exponentiel par rapport au nombre de composants du modèle. Par exemple, un modèle contenant 100 variables binaires peut présenter $2^{100} > 10^{30}$ états possibles. Un tel graphe n'est donc pas explicitement calculable pour de grands réseaux, mais il constitue un objet d'étude central pour les modèles qualitatifs. Il sert notamment de support pour la définition de propriétés dynamiques de modèles entre les différents formalismes. L'exemple 1.3 représente les différents graphes de transition du réseau booléen de l'exemple 1.1 avec trois sémantiques usuelles : synchrone, asynchrone et asynchrone généralisée. La notion

de graphe de transitions sera réutilisée dans le chapitre 2 dans le cadre de la résolution de problèmes d'analyse de la dynamique, ainsi que dans le chapitre 3 pour l'inférence des modèles.

Exemple 1.3. *La figure 1.4 illustre les trois graphes de transitions correspondants aux sémantiques synchrone (T_{sync}), asynchrone (T_{async}) et asynchrone généralisée (T_{gen}). On remarque bien que $T_{gen} \supset T_{sync} \cup T_{async}$. On peut également vérifier le caractère non déterministe des réseaux booléens asynchrones et asynchrones généralisés puisque certains sommets admettent plusieurs arcs sortants pour ces deux sémantiques.*

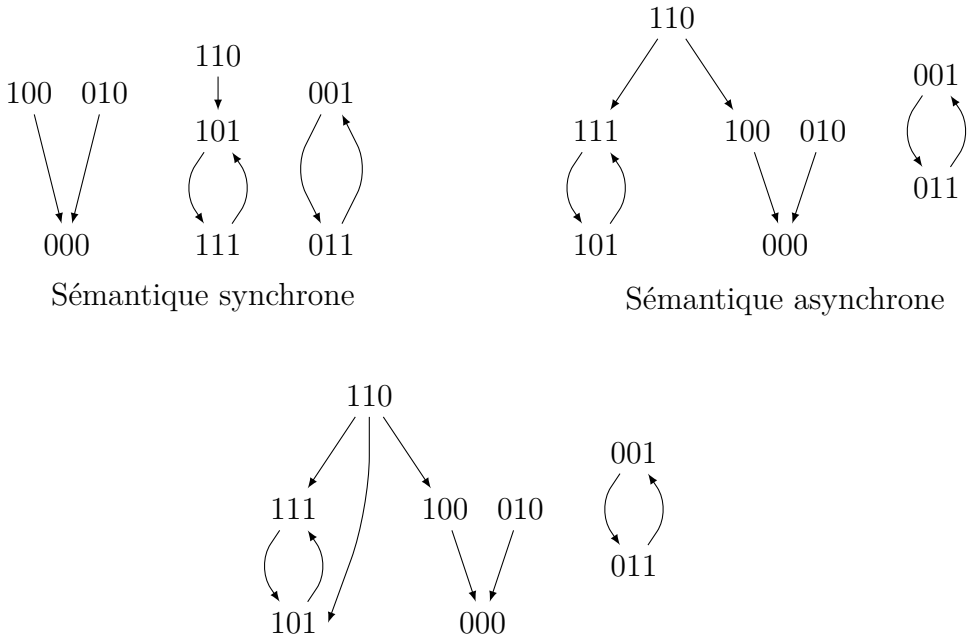

FIGURE 1.4 – Le graphe de transitions du réseau booléen de l'exemple 1.1 pour trois sémantiques usuelles : asynchrone, synchrone et asynchrone généralisée.

Propriétés sur les graphes de transitions

À défaut d'être calculé entièrement, le graphe de transitions d'un modèle qualitatif est généralement utilisé pour caractériser certains comportements définis à partir de propriétés formelles. Par exemple, l'analyse des états stables (*stable states* ou *steady states*) consiste à rechercher les états dans lesquels le système est considéré comme étant stable, c'est-à-dire que ses composants n'évoluent plus. Du point de vue biologique, l'analyse des

états stables permet la validation d'un modèle par comparaison aux observations expérimentales, et s'apparente à des propriétés importantes comme le type d'une cellule dans un processus de différenciation (Mori et al., 2017). Dans le cas des réseaux booléens synchrones, les états stable sont considérés comme étant les points fixes (Campbell et al., 2014), c'est-à-dire les états pour lesquels l'unique transition du modèle le conduit dans ces mêmes états (Yi Ming Zou, 2013). Pour certains modèles non déterministes comme les réseaux d'automates, les points fixes sont définis comme étant les états qui n'admettent pas de successeurs dans le graphe de transitions du modèle (Ben Abdallah et al., 2017b). Un autre problème souvent étudié est l'accessibilité (ou atteignabilité), qui consiste à déterminer s'il existe un chemin entre un état initial et un état cible dans le graphe de transitions. Du point de vue biologique, ce problème permet par exemple de vérifier l'activation d'un facteur de transcription suite à une perturbation du réseau (Paulevé, 2018). L'accessibilité est aussi reliée à l'étude des *cut sets*, c'est-à-dire d'ensembles d'états locaux nécessaires pour la réalisation d'une accessibilité (Paulevé, 2017). Enfin, un dernier problème majeur est l'analyse des attracteurs d'un modèle. Un attracteur désigne un ensemble d'états dans lesquels un modèle peut évoluer, mais à partir desquels il ne peut s'échapper. Les attracteurs permettent de caractériser le comportement à long terme d'un modèle qualitatif. D'un point de vue biologique, ils peuvent notamment être interprétés comme les types cellulaires (Shen-Orr et al., 2010) et permettent de caractériser des processus de différenciation ou d'adaptation du système (Chatain et al., 2014).

1.3 Les enjeux de la modélisation qualitative

Comme nous l'avons souligné précédemment, la modélisation qualitative est un outil puissant pour l'étude de systèmes biologiques. Cependant, les modèles qualitatifs sont des objets complexes à manipuler, et plusieurs défis se présentent concernant leur utilisation dans la recherche biologique et biomédicale. Nous séparons ici ces enjeux dans deux catégories principales qui sont traitées dans cette thèse. En premier lieu, l'**analyse** des modèles qualitatifs, abordée au chapitre 2, a pour but de proposer des méthodes efficaces pour l'étude des propriétés de modèles qualitatifs. En second lieu, le problème de l'**inférence** de modèles, abordé dans les chapitres 3 et 4, consiste à déterminer comment concevoir un modèle qualitatif d'un système biologique. On s'intéresse dans cette thèse en particulier à une conception automatique, basée principalement sur des données expérimentales.

1.3.1 L'analyse des modèles qualitatifs

Une question importante qui se pose lorsqu'un modèle qualitatif est établi pour un système biologique donné est la manière d'exploiter ce modèle afin de contribuer aux connaissances biologiques sur le système étudié. Une première étape consiste à s'assurer que le modèle représente le système réel de manière pertinente. Pour vérifier cela, il est possible de le confronter aux connaissances du système dans la littérature, ainsi qu'aux autres modèles existants. Il peut être également intéressant d'effectuer des expérimentations directement sur le système étudié mais celles-ci peuvent être coûteuses et hors de portée. Dans le cas où le modèle ne décrit pas correctement le système réel, les incohérences observées peuvent être exploitées afin d'améliorer les connaissances sur le système ainsi que la modélisation. Enfin, quand un modèle est cohérent avec le système étudié, il est intéressant de l'utiliser afin de générer des comportements encore non observés. Ces comportements peuvent par exemple être utilisés pour prédire la réponse du système à une perturbation. Une question transverse est celle de la reprogrammation des modèles, c'est-à-dire la modification de ceux-ci afin d'en changer certains de leurs comportements (Mandon et al., 2019).

Les propriétés des modèles qualitatifs mentionnées précédemment permettent ainsi de caractériser les comportements des modèles dans le but de les exploiter plus facilement. Par ailleurs, ces propriétés sont difficiles à calculer car elles correspondent à des problèmes algorithmiques dont la complexité peut être *NP-Complet* ou encore *PSPACE-Complet*. Pour cette raison, des méthodes efficaces sont nécessaires afin de rendre possible l'analyse de modèles de grande taille. Comme mentionné précédemment, l'analyse des modèles bénéficie des travaux développés en informatique, en particulier dans le domaine des méthodes formelles. Dans un premier temps, il est donc intéressant de se baser sur des méthodes existantes, notamment développées pour la vérification formelle. Cependant, les méthodes générales sont souvent moins efficaces pour traiter un problème précis, et on peut s'attendre à ce que des méthodes dédiées permettent de résoudre le problème plus facilement. Dans le chapitre 2, nous nous concentrons sur le problème précis et formellement identifié de l'analyse de l'accessibilité dans les réseaux d'automates asynchrones, et nous développons une nouvelle méthode pour ce problème afin d'en améliorer la résolution. Notre approche peut être assimilée à celle généralement adoptée en recherche opérationnelle, où les problèmes d'optimisation sont formalisés et étudiés très spécifiquement afin d'en améliorer le plus possible le pouvoir de résolution.

1.3.2 L'inférence des modèles qualitatifs

Une autre question importante sur la modélisation qualitative concerne la conception d'un modèle cohérent pour un système donné. Les premiers modèles qualitatifs ont été élaborés à partir de connaissances de la littérature biologique, élaborée elle-même à partir d'observations expérimentales. Par exemple, la méthode d'identification de modèle à partir de réseaux booléens aléatoires mentionnée précédemment a été basée sur un réseau de régulation déterminé expérimentalement dans (Lee, 2002), et une méthode basée également sur la littérature a été utilisée pour l'élaboration d'un modèle logique du bactériophage lambda dans (Thieffry et al., 1995).

L'approche de conception manuelle est cependant limitée car elle nécessite une certaine expertise sur le système afin de pouvoir en comprendre la littérature et elle est chronophage, et donc limitée à de petits modèles. De plus, le développement de nouvelles technologies expérimentales pour l'étude des systèmes biologiques conduit à l'obtention de très grandes quantités de données, qui sont de plus en plus précises. Pour ces raisons, des approches automatiques sont apparues pour inférer les modèles à partir de données variées. Certaines approches comme celle proposée dans (Khalis et al., 2009) utilisent comme données de départ un graphe d'influence ainsi que des propriétés dynamiques et se concentrent sur l'identification de paramètres de modèles qualitatifs tels que les réseaux de Thomas. D'autres approches se basent au contraire uniquement sur des données expérimentales, comme c'est le cas dans (Lim et al., 2016). Dans cette thèse, nous adoptons cette dernière approche pour l'inférence des modèles, en l'inscrivant dans une démarche d'apprentissage automatique. Plusieurs avantages se présentent pour l'utilisation exclusive de données expérimentales. Cela permet d'éviter notamment l'utilisation de données hétérogènes dans un même modèle, qui peuvent parfois ne pas être compatibles. De plus, la séparation entre un modèle inféré à partir de données expérimentales d'une part, et les connaissances théoriques d'autre part permet de faciliter leur confrontation. En effet, l'utilisation de connaissances théoriques sur un système risque de biaiser le modèle et de ne pas faire pleinement ressortir les informations présentes dans les données. Enfin, en se concentrant sur des données expérimentales, un des objectifs est de proposer des méthodes robustes à l'aspect bruité et incomplet (*sparse*) de ces données qui complexifient la modélisation abstraite. Dans le chapitre 3, nous nous concentrons sur les différentes méthodes d'inférence de paramètres et d'apprentissage automatique des modèles, et nous proposons une nouvelle méthode d'apprentissage automatique spécifiquement pensée pour la modéli-

sation qualitative à partir de données expérimentales. Dans le chapitre 4, nous appliquons notre méthode sur des données expérimentales de séquençage *single-cell*. Ce dernier chapitre se concentre en particulier sur l'exploitation de tels modèles expérimentaux, en confrontation aux données d'origine. On s'attache notamment à vérifier rigoureusement que la modélisation proposée ne commet pas d'*overfitting*.

Chapitre 2

Méthode formelle pour l'analyse de modèles qualitatifs

Résumé

Dans ce chapitre, nous abordons le problème de l'analyse des modèles qualitatifs en biologie des systèmes. On se concentre en particulier sur les différentes méthodes algorithmiques permettant de calculer les propriétés de modèles qualitatifs. Nous étudions en particulier le problème d'accessibilité dans les réseaux d'automates asynchrones. Ce problème est en effet central dans l'étude du comportement dynamique des modèles qualitatifs, et les méthodes de résolution existantes restent inefficaces sur certaines instances. Nous proposons ainsi une nouvelle méthode basée sur la combinaison des approches de *Bounded Model Checking* avec une méthode reposant sur l'analyse statique, issue de travaux antérieurs dans les réseaux d'automates.

2.1 Analyse formelle de la dynamique des réseaux

Les modèles discrets qualitatifs étant des objets complexes à étudier, différentes propriétés ont été formalisées à partir de leur graphe de transitions afin de caractériser plus facilement les comportements des systèmes biologiques modélisés (voir chapitre 1). Les problèmes algorithmiques associés au calcul de ces propriétés sont par ailleurs souvent difficiles à résoudre. Bien que les solutions à ces problèmes soient généralement énumérables (lorsque les modèles sont finis), les méthodes d'exploration naïve conduisent à des algorithmes dont la complexité temporelle est au minimum exponentielle par rapport à la taille

du modèle. Par exemple, les problèmes du calcul des points fixes et de certains attracteurs sont généralement classifiés *NP-Difficile* (Akutsu et al., 2012 ; Zhang et al., 2007). Le problème d'accessibilité est quant à lui généralement classifié *PSPACE-Complet* puisque la longueur des chemins d'accessibilité n'est pas forcément polynomiale en le nombre de composants des modèles (Paulevé, 2018). Face à la difficulté de ces problèmes, les méthodes proposées pour leur résolution s'appuient sur différents champs de recherche en informatique. Il existe en effet une forte proximité entre la modélisation qualitative des réseaux biologiques et l'étude formelle des systèmes informatiques et électroniques. C'est la raison pour laquelle les méthodes introduites dans le domaine du *Model Checking* (c'est-à-dire la vérification formelle de modèles ou de programmes (Clarke, 2008)) sont appliquées à la résolution des problèmes mentionnés précédemment. En particulier, les techniques de *Bounded Model Checking* (Biere et al., 2009) et d'analyse statique (Wichmann et al., 1995) ont été utilisées à cette fin comme nous le renseignons dans la partie suivante.

Malgré les efforts développés pour la résolution des problèmes d'analyse, le calcul des propriétés dynamiques reste difficile sur les modèles de grande taille. Une étude approfondie de ces problèmes est donc nécessaire afin de développer de nouvelles méthodes pour en améliorer la capacité de résolution. Dans ce but, nous nous concentrons dans la suite de ce chapitre sur un problème particulier : l'accessibilité dans les réseaux d'automates asynchrones. Dans la partie suivante, nous motivons ce choix et nous donnons une définition formelle de ce problème. L'identification des limites des méthodes de résolution existantes nous permet alors de proposer une nouvelle méthode, à partir de la combinaison de deux méthodes existantes.

2.2 Le problème d'accessibilité dans les réseaux d'automates

Les réseaux d'automates, que nous avons mentionnés dans la partie 1.2.1 du chapitre 1, sont intéressants pour la modélisation qualitative car ils permettent de représenter intuitivement les interactions entre composants biologiques multivalués via des transitions locales. Nous nous intéressons en particulier aux réseaux d'automates asynchrones, dans lesquels un seul automate peut être mis à jour par transition du modèle. Cette sémantique a en effet reçu un certain intérêt dans l'étude des modèles qualitatifs (Assmann et al., 2009). Le calcul de l'accessibilité est important car il permet d'en analyser le comporte-

ment dynamique en profondeur. Par exemple, on peut l'utiliser pour vérifier la cohérence d'un modèle par rapport à des observations ou des connaissances a priori du système, et pour le réviser en conséquence. Comme nous l'avons mentionné dans le chapitre 1, une telle analyse peut également servir à raffiner le comportement d'un modèle sur-approximant un système donné. La sémantique synchrone représente par ailleurs un défi en termes de résolution car, comme nous l'avons mentionné dans le chapitre 1, celle-ci engendre un grand nombre de comportements dans un modèle et pose ainsi des difficultés aux outils de vérification formelle existants. De même que pour d'autres formalismes qualitatifs, le problème d'accessibilité dans les réseaux d'automates est ainsi catégorisé *PSPACE-Complet* et des méthodes avancées sont nécessaires pour sa résolution dans des modèles de grande taille. Les différentes approches de résolution existantes comportent ainsi toutes des limites qui les rendent inefficaces sur certaines instances du problème.

Dans la suite de cette partie, nous définissons formellement les réseaux d'automates asynchrones ainsi que le problème d'accessibilité qui y est associé. Les méthodes existantes pour la résolution de l'accessibilité dans les réseaux d'automates sont également détaillées, et leurs limites sont mises en évidence afin de justifier notre nouvelle approche de résolution. Nous définissons enfin les outils d'analyse statique développés pour la résolution de l'accessibilité. Ces outils sont réutilisés dans la partie 2.3.2 pour la mise en place de notre méthode de résolution.

2.2.1 Formalisation du problème d'accessibilité

Les réseaux d'automates sont définis à partir de plusieurs automates, représentant les différents composants du modèle (par exemple gène, protéine, etc.). Les automates comportent plusieurs états locaux, permettant de modéliser plusieurs niveaux d'activation des composants. Un état global du modèle est représenté par un état local pour chaque automate du système, et le modèle possède ainsi un nombre fini d'états qui est exponentiel par rapport au nombre d'automates. Les interactions entre les composants du modèle sont représentées par des transitions locales à l'intérieur de chaque automate. Ces transitions modélisent le passage d'un niveau d'activité d'un composant vers un autre, conditionné par la présence de certains niveaux d'activités pour un ou plusieurs autres automates. Une transition est ainsi possible uniquement si les conditions des autres automates sont vérifiées dans l'état à partir duquel la transition est jouée. La définition suivante formalise les différents éléments permettant de définir un réseau d'automates.

Définition 2.1. *Réseau d'automates*
Un réseau d'automates est défini par un triplet $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *où :*

- $\Sigma = \{a, b, c, ..\}$ *est l'ensemble fini des automates.*
- $\mathcal{S} = \prod_{x \in \Sigma} \mathcal{S}_x$ *est l'ensemble fini des états, où* $\mathcal{S}_x = \{x_0, x_1, ..\}$ *est l'ensemble fini des états locaux de l'automate* $x \in \Sigma$*.*
- $\mathcal{T}_x \subset \{(x_i, x_j, l) \mid x_i, x_j \in \mathcal{S}_x, x_i \neq x_j, l \subset \bigcup_{y \in \Sigma \setminus \{x\}} \mathcal{S}_y \text{ et } |l \cap \mathcal{S}_y| \leq 1 \quad \forall y \in \Sigma\}$ *est l'ensemble des transitions de l'automate* x*, et* $\mathcal{T} = \bigcup_{x \in \Sigma} \mathcal{T}_x$ *est l'ensemble des transitions du réseau.*

Étant donné un état global $s \in \mathcal{S}$, l'état local de l'automate x est noté $s(x)$. Une transition $\tau \in \mathcal{T}_x$ est représentée par un triplet $(x_i, x_j, \{y_k, z_l, ..\})$ où x_i, l'état local origine, est noté $orig(\tau)$, x_j, l'état destination, est noté $dest(\tau)$, et l'ensemble des états locaux $\{y_k, z_l, ..\}$ qui doivent être vérifiés pour jouer la transition est noté $enab(\tau)$. La notation $x_i \xrightarrow{y_k, z_l} x_j$ est utilisée pour représenter la transition τ. Enfin, les transitions bouclant sur le même état local, c'est-à-dire de la forme $x_i \to x_i$, ne sont pas considérées ici. En effet, les transitions d'un réseau d'automates visent à représenter un changement dans le modèle, et l'absence de changement sur un automate est vérifiée quand aucune transition n'est jouée.

Exemple 2.1. *La figure 2.1 présente un réseau d'automates* $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *comportant 3 automates. Dans ce modèle,* $\Sigma = \{a, b, c\}$ *et* $\mathcal{S} = \{(a_i, b_j, c_k) \mid i, j, k \in [\![0, 2]\!]\}$*. Dans l'automate a,* $\mathcal{S}_a = \{a_0, a_1, a_2\}$ *et* $\mathcal{T}_a = \{a_0 \xrightarrow{b_2} a_1, a_1 \xrightarrow{b_1, c_2} a_2\}$*. Pour la transition* $\tau = a_1 \xrightarrow{b_1, c_2} a_2$*, on a* $orig(\tau) = a_1$*,* $dest(\tau) = a_2$ *et* $enab(\tau) = \{b_1, c_2\}$*. Enfin, on remarque que certaines transitions n'ont pas de conditions. Par exemple, c'est le cas de* $b_0 \to b_1$*, et donc dans ce cas* $enab(b_0 \to b_1) = \varnothing$*.*

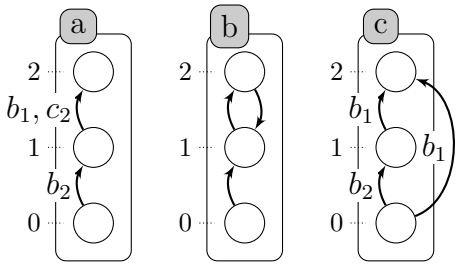

FIGURE 2.1 – Un réseau d'automates avec 3 automates.

Comme indiqué précédemment, nous employons la sémantique asynchrone des réseaux d'automates, ce qui signifie qu'à partir d'un état global, une seule transition parmi toutes celles qui sont possibles peut être appliquée. La sémantique asynchrone rend le modèle non déterministe au sens où chaque transition jouable à partir d'un état donné conduit à un état différent. Cette précision permet de définir le passage d'un état à un nouvel état dans le modèle, ce qui se traduit ici par la **jouabilité** d'une transition. On peut aussi noter que puisque les transitions entre les mêmes états locaux ($x_i \to x_i$) ne sont pas considérées, les boucles de longueur 1 ne peuvent pas apparaître dans le graphe de transitions d'un réseau d'automates.

Définition 2.2. *Jouabilité d'une transition (sémantique asynchrone)*
Étant donné un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *:*

- *Une transition* $\tau \in \mathcal{T}_x$ *est jouable à partir d'un état* $s \in \mathcal{S}$ *si* $s(x) = orig(\tau)$ *et* $\forall y_i \in enab(\tau), y_i = s(y)$*. On définit le prédicat jouable tel que* $jouable(s, \tau) = \top$ *si et seulement si* τ *est jouable à partir de* s*.*
- *L'opérateur de jouabilité, noté* $\cdot$*, est défini par* $s \cdot \tau = s'$ *où* $s'(y) = s(y) \quad \forall y \in \Sigma \setminus \{x\}$ *et* $s'(x) = dest(\tau)$ *si* $jouable(s, \tau) = \top$*. Si* $jouable(s, \tau) = \bot$*,* $s \cdot \tau$ *n'est pas défini.*
- *Une séquence* π *de transitions de longueur* n *est notée* $\pi = (\pi^1 :: \pi^2 :: .. :: \pi^n)$ *avec* $\pi^i \in \mathcal{T} \quad \forall i$ *et jouer* π *(lorsque c'est possible) est noté* $s \cdot \pi = s \cdot \pi^1 \cdot \pi^2 \cdot .. \cdot \pi^n$*, avec* $s \in \mathcal{S}$*.*

Exemple 2.2. *Dans le réseau d'automates de la figure 2.1, à partir de l'état* $s = (a_0, b_2, c_0)$*, les transitions* $\tau_1 = a_0 \xrightarrow{b_2} a_1$ *et* $\tau_2 = c_0 \xrightarrow{b_2} c_1$ *sont jouables et la transition* $\tau_3 = b_1 \to b_2$ *ne l'est pas. Dans ce modèle,* $s \cdot \tau_1 = (a_1, b_2, c_0)$*. Pour la séquence* $\pi = (c_0 \xrightarrow{b_2} c_1 :: b_2 \to b_1)$*,* $s \cdot \pi = s \cdot \pi^1 \cdot \pi^2 = (a_0, b_1, c_1)$*.*

Le problème d'accessibilité dans les réseaux d'automates peut être défini par l'existence d'une séquence de transitions qui permet au modèle d'évoluer d'un état initial à un état cible vérifiant la présence de certains états locaux. Le problème peut être réduit à l'accessibilité d'un unique état local en utilisant un automate supplémentaire dans le modèle, comme mentionné dans (Folschette et al., 2015b). Pour cette raison, nous ne considérons que ce dernier cas ici. Nous formalisons ainsi le problème dans la définition suivante.

Définition 2.3. *Accessibilité dans les réseaux d'automates asynchrones*

- *Entrée : Un réseau d'automates* $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$*, un couple d'accessibilité* $\mathcal{R} = (s_0, x_i)$ *avec* $s_0 \in \mathcal{S}$ *un état initial et* $x_i \in \mathcal{S}_x$ *un état local objectif*
- *Sortie :* $\top$ *ou* $\bot$
- *Question : Existe-t-il une séquence de transitions* $\pi = (\pi^1 :: \pi^2 :: .. :: \pi^n)$ *telle que* $s_0 \cdot \pi = s'$ *avec* $s'(x) = x_i$ *?*

Concernant la complexité, il est possible de montrer que le problème appartient à la classe de complexité *PSPACE-Complet* à partir d'une traduction des réseaux d'automates asynchrones en réseaux de *Petri* bornés (Paulevé, 2018). Il est en effet établi que la vérification de l'accessibilité dans les réseaux de *Petri* appartient également à la classe *PSPACE-Complet* (Cheng et al., 1995). De manière intuitive, le nombre d'états d'un réseau d'automates est fini et exponentiel par rapport au nombre d'automates et le problème d'accessibilité peut donc être vu comme la recherche d'un chemin entre deux états dans le graphe de transitions du modèle, ce qui permet notamment une énumération naïve des solutions.

À partir de la définition 2.3 de l'accessibilité, une solution d'une instance de ce problème est définie comme une séquence de transitions réalisant l'accessibilité, comme formalisé dans la définition qui suit. Par ailleurs, une solution peut être arbitrairement longue et contenir des transitions non nécessaires, c'est-à-dire des transitions pouvant être retirées tout en conservant la validité de la solution pour la question d'accessibilité. Pour cela, nous définissons également la notion de sous-solution, correspondant à une sous-séquence d'une solution qui conserve sa validité. Enfin, nous définissons la notion de solution minimale, qui correspond à une solution n'admettant pas de sous-solution. Ces notions seront réutilisées par la suite pour la résolution de l'accessibilité.

Définition 2.4. *Solutions, sous-solutions et solutions minimales d'une accessibilité*
Soit $\mathcal{R} = (s_0, x_i)$ *une instance d'accessibilité.*
- Une séquence π_1 *de transitions est une solution pour l'instance* $\mathcal{R}$ *si* $[s_0 \cdot \pi_1](x) = x_i$*.*
- Une séquence π_2 *de transitions est une sous-solution d'une solution* π_1 *si* π_2 *est une solution et si il existe une injection* $\phi : [\![1, |\pi_2|]\!] \to [\![1, |\pi_1|]\!]$ *telle que* $\forall i \in [\![1, |\pi_2|]\!], \pi_1^{\phi(i)} = \pi_2^i$ *et* $\forall j \in [\![1, |\pi_2| - 1]\!], \phi(j) < \phi(j+1)$ *(*π_2 *contient un sous-ensemble des transitions de* π_1*, dans le même ordre).*
- Une solution π *est une solution minimale pour* $\mathcal{R}$ *s'il n'existe pas de sous-solution stricte* π' *de* π*.*

Exemple 2.3. *Dans le réseau d'automates de la figure 2.1, la réponse du problème d'accessibilité* $\mathcal{R} = ((a_0, b_0, c_0), a_2)$ *est* $\top$.
$\pi_1 = (b_0 \to b_1 :: b_1 \to b_2 :: b_2 \to b_1 :: b_1 \to b_2 :: a_0 \xrightarrow{b_2} a_1 :: b_2 \to b_1 :: c_0 \xrightarrow{b_1} c_2 :: a_1 \xrightarrow{b_1,c_2} a_2)$ *est une solution (*$[(a_0, b_0, c_0) \cdot \pi_1](a) = a_2$*).*
La solution $\pi_2 = (b_0 \to b_1 :: b_1 \to b_2 :: a_0 \xrightarrow{b_2} a_1 :: b_2 \to b_1 :: c_0 \xrightarrow{b_1} c_2 :: a_1 \xrightarrow{b_1,c_2} a_2)$ *est une sous-solution de* π_1 *et est une solution minimale.*

2.2.2 Méthodes de résolution de l'accessibilité et limites

Comme indiqué précédemment, plusieurs méthodes de résolution ont été proposées pour traiter le problème d'accessibilité dans les réseaux d'automates. Tout d'abord, la forte proximité entre les formalismes de modélisation qualitative tels que les réseaux d'automates et les modèles utilisés dans les domaines de la vérification formelle et du *Model Checking* conduisent à réutiliser les outils déjà développés pour ces derniers. Par exemple, cette approche a été employée pour l'analyse de modèles biologiques dans (Gallet et al., 2014 ; Klarner et al., 2012a ; Klarner et al., 2012b). Cette méthode a également été utilisée pour la vérification des réseaux biochimiques (Češka et al., 2014), ainsi que dans le cas de la vérification de l'analyse des réseaux de signalisation (Heath et al., 2008). Les méthodes de résolution symbolique telles que les Diagrammes de Décision Binaires étaient notamment fréquentes en *Model Checking* (Burch et al., 1990), et ces méthodes sont encore utilisées dans des outils de vérification formelle génériques tels que *NuSMV* [1] (Cimatti et al., 1999) et *ITS-tools* [2] (Thierry-Mieg, 2015). Bien que ce type de méthode a l'avantage d'être applicable sur beaucoup de formalismes différents, la résolution n'est pas toujours efficace sur des modèles de grande taille (Bryant, 1986). Il a notamment été montré que ces méthodes ne sont pas avantageuses sur les réseaux d'automates en comparaison à des approches plus spécifiques à ces modèles (Paulevé, 2017).

Devant la complexité des problèmes traités en *Model Checking* et devant la taille importante des modèles à analyser, l'approche du *Bounded Model Checking* (*BMC*) est devenue incontournable pour la vérification de modèles, notamment dans les applications industrielles (Biere et al., 2009). Cette approche consiste à encoder (ou encore traduire) une séquence d'exécution du modèle de longueur fixée dans un langage logique en ajoutant des contraintes modélisant une propriété à vérifier. Un paramètre k déterminant la

1. http://nusmv.fbk.eu/NuSMV/
2. https://lip6.github.io/ITSTools-web/index.html

longueur de la séquence doit donc être décidé à l'avance. L'utilisation d'un algorithme de résolution pour le langage choisi (généralement appelé *solveur*) permet ensuite de vérifier parmi toutes les séquences d'exécution du modèle de longueur k celles qui vérifient la propriété recherchée. Cette approche est notamment utilisée pour rechercher des contre-exemples permettant d'infirmer certaines propriétés, et donc de démontrer qu'un système n'est pas sûr. Le *Bounded Model Checking* a été appliqué à plusieurs reprises pour la vérification des modèles qualitatifs en biologie des systèmes (Dubrova et al., 2011 ; Nabli et al., 2016). Les solveurs *SAT*, visant à résoudre une instance de satisfaisabilité en logique propositionnelle, sont notamment très présents en *Bounded Model Checking*. Ils ont par exemple été utilisés pour traiter le problème d'accessibilité dans des réseaux de Petri (Ogata et al., 2004), ainsi que pour le calcul des attracteurs dans les réseaux booléens asynchrones (Van Giang et al., 2020) (dans ce dernier, l'approche utilisée ne suit toutefois pas complètement le schéma du *Bounded Model Checking*). Un encodage *SMT* a également été proposé pour la résolution de l'accessibilité dans des réseaux de régulation génétiques synchrones (Giacobbe et al., 2017). Concernant les réseaux d'automates, le *Bounded Model Checking* a été utilisé dans (Ben Abdallah et al., 2017b) pour calculer les attracteurs à partir d'un encodage avec le langage *Answer Set Programming* (*ASP*). Bien que les solveurs *SAT* et *ASP* soient très performants et permettent de traiter des modèles avec plusieurs centaines, voire milliers de composants, le *Bounded Model Checking* présente des limitations importantes dans la résolution théorique de l'accessibilité. Il est en effet nécessaire de prévoir la longueur k des séquences encodées à partir du modèle. On peut ainsi prouver que l'accessibilité est possible lorsque le solveur trouve une solution, mais l'absence de solution permet uniquement de prouver que l'accessibilité n'est pas possible pour les séquences de longueur $\leq k$, et il reste une incertitude pour les séquences plus longues. Dans le cas d'un modèle fini comme les réseaux d'automates, le choix du nombre d'états du modèle pour k est suffisant pour fournir une preuve de l'inaccessibilité mais ce nombre est exponentiel en le nombre d'automates et n'est donc pas utilisable en pratique. Plusieurs méthodes ont été proposées pour choisir un paramètre k assurant la complétude, parfois appelé *completeness threshold*, permettant donc de prouver en l'absence de solution que les séquences de longueur supérieure ne réalisent pas non plus l'accessibilité. Par exemple, la technique de *k-induction* consiste à déterminer la longueur maximale des chemins minimaux qui atteignent l'état local cible de l'accessibilité (Biere et al., 2009). Ce type de méthode reste cependant général et n'est pas forcément efficace dans les réseaux d'automates, le k résultant pouvant être grand.

Une approche de vérification formelle dédiée spécifiquement aux réseaux d'automates, basée sur l'analyse statique et inspirée de l'interprétation abstraite (Cousot et al., 1977) a été proposée pour résoudre l'accessibilité. L'analyse statique vise à vérifier les propriétés d'un modèle sans l'exécuter (ou encore le simuler), et donc sans procéder à une exploration exhaustive de son graphe de transitions (Wichmann et al., 1995). Cette approche basée sur l'analyse statique a été présentée dans (Folschette et al., 2015b) et permet de démontrer l'accessibilité à partir de la vérification de conditions nécessaires et suffisantes pour cette propriété. La vérification de ces conditions peut être faite en temps polynomial par rapport à la taille du modèle. Plus précisément, cette méthode permet de prouver que l'accessibilité est réalisable lorsque les conditions suffisantes sont vérifiées, et qu'elle ne l'est pas lorsque les conditions nécessaires ne sont pas vérifiées. Elle a notamment fait l'objet d'une implémentation dans l'outil *Pint*[3] (Paulevé, 2017) dédié à l'analyse des réseaux d'automates. Bien que cette approche soit très efficace, elle ne permet cependant pas toujours de conclure. En effet, lorsque les conditions nécessaires sont vérifiées et que les conditions suffisantes ne le sont pas, cette méthode n'apporte aucune réponse.

Malgré l'existence de différentes approches de résolution, certaines instances de problème d'accessibilité sont impossibles à résoudre. Afin de les identifier, les différents cas se présentant pour une instance d'accessibilité ont été représentés en figure 2.2. L'axe horizontal de cette figure tente de classer les instances par l'absence (à gauche, cas 1 et 2) ou l'existence (à droite, cas 3 et 4) d'une solution. La difficulté de l'instance est également représentée par sa distance par rapport au milieu de l'axe. On peut ainsi supposer que les cas les plus simples, que nous plaçons aux extrémités, sont les cas qui peuvent être traités avec les outils d'analyse statique, c'est-à-dire le cas 1 où les conditions nécessaires pour l'accessibilité ne sont pas vérifiées, permettant de prouver l'absence de solution, et le cas 4 où les conditions suffisantes sont vérifiées, permettant de prouver l'existence d'une solution. Concernant le cas 3, on peut supposer que l'existence d'une solution est efficacement traitable avec les approches en *Bounded Model Checking*. En effet, dès lors qu'une solution existe, et que sa longueur est raisonnable, la puissance des solveurs logiques permet de la trouver assez facilement quand le paramètre k est adapté. Toutefois, si les solutions pour une telle instance sont particulièrement longues, il peut être difficile de les calculer avec cette approche, ce qui est représenté par l'arc de cercle sur la figure afin d'exclure

3. https://loicpauleve.name/pint/

une partie des instances admettant une solution. Le cas 2 restant est celui où il n'existe pas de solutions et que les conditions nécessaires sont tout de même vérifiées. Ce sont les instances qui sont visées par la méthode proposée dans ce chapitre. Pour cela, nous proposons un encodage en *Bounded Model Checking*, combiné à une propriété formelle afin de garantir l'exhaustivité de la recherche de solution avec un paramètre k adéquat, qui est obtenu à partir d'une nouvelle extension des outils d'analyse statique formalisés dans la partie suivante. La difficulté de cette approche réside donc dans l'obtention d'une telle propriété formelle garantissant un paramètre k de taille raisonnable.

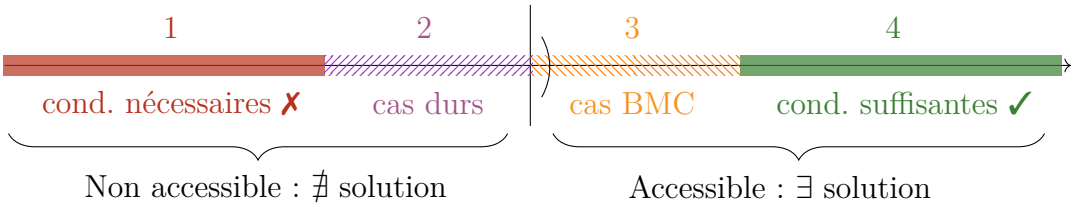

FIGURE 2.2 – Illustration figurative des différents cas possibles pour une instance d'accessibilité, en confrontation aux méthodes de résolution proposées.

2.2.3 Analyse statique dans les réseaux d'automates

Les outils d'analyse statique développés dans (FOLSCHETTE et al., 2015b) sont basés sur une résolution locale de l'accessibilité, c'est-à-dire en ne considérant qu'un automate et sans prendre en compte les conditions des transitions dans un premier temps. Cette accessibilité locale peut être résolue rapidement par énumération, et elle peut être utilisée pour obtenir des informations intéressantes pour la résolution d'une accessibilité globale. En effet, l'absence de solution locale permet déjà de prouver l'absence de solution pour une accessibilité. Dans le cas où il existe une ou plusieurs solutions locales pour une accessibilité locale, il est également possible de définir récursivement des sous-problèmes d'accessibilité locaux selon les conditions des transitions présentes dans les solutions locales précédemment identifiées. Cette décomposition récursive du problème a été utilisée pour définir un graphe contenant les solutions locales, et dans lequel la vérification de certaines propriétés permet de démontrer dans certains cas qu'il existe une solution au problème d'accessibilité, ou qu'il n'en existe pas. La définition suivante introduit la notion d'objectif, qui correspond à un problème d'accessibilité locale, et celle de solution locale, correspondant aux solutions d'un objectif.

Définition 2.5. *Objectif et solution locale*
Étant donné un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *:*

- *Un problème d'accessibilité locale dans un automate* $x \in \Sigma$ *est un couple* $(x_i, x_j) \in \mathcal{S}_x \times \mathcal{S}_x$ *appelé objectif, et noté* $x_i \rightsquigarrow x_j$
- *Une solution locale d'un objectif* $x_i \rightsquigarrow x_j$ *est une séquence acyclique de transitions* $\eta = (\eta^1 :: .. :: \eta^n)$ *(avec* $\eta^i \in \mathcal{T}_x \quad \forall i \in [\![1, n]\!]$*) telle que* $orig(\eta^1) = x_i$*,* $dest(\eta^n) = x_j$ *et* $dest(\eta^k) = orig(\eta^{k+1}) \quad \forall k \in [\![1, n-1]\!]$ *et* $\forall i, j \in [\![1, n]\!], j > i \implies orig(\eta^i) \neq dest(\eta^j)$
- *L'ensemble fini de toutes les solutions locales associées à* $x_i \rightsquigarrow x_j$ *est noté* $lpaths(x_i \rightsquigarrow x_j)$
- $lpaths(x_i \rightsquigarrow x_i) = \{\varepsilon\} \quad \forall x_i \in \mathcal{S}_x, \forall x \in \Sigma$

Exemple 2.4. *Dans le réseau d'automates de la figure 2.1 en page 42, l'objectif* $c_0 \rightsquigarrow c_2$ *admet deux solutions locales. Une première solution :* $\eta_1 = (c_0 \xrightarrow{b_2} c_1 :: c_1 \xrightarrow{b_1} c_2)$*, et une seconde :* $\eta_2 = (c_0 \xrightarrow{b_1} c_2)$*. On a donc* $lpaths(c_0 \rightsquigarrow c_2) = \{\eta_1, \eta_2\}$*. L'objectif* $c_2 \rightsquigarrow c_2$ *n'admet qu'une solution triviale* $\eta_3 = \varepsilon$ *puisque l'accessibilité locale est déjà satisfaite, ainsi* $lpaths(c_2 \rightsquigarrow c_2) = \{\eta_3\} = \{\varepsilon\}$*.*

Afin d'obtenir un maximum d'informations sur la résolution d'un problème d'accessibilité, il est possible de lier des objectifs entre eux et d'étudier leurs potentielles combinaisons pour former une solution. La définition suivante introduit le Graphe de Causalité Locale (que nous désignerons également par GCL par la suite) qui est l'outil central pour l'analyse de l'accessibilité et qui est basé sur la connexion de différents objectifs d'une instance d'accessibilité. Il contient l'objectif qui lie l'état initial à l'état local cible de l'accessibilité, ainsi que plusieurs objectifs intermédiaires, capturant de manière **suffisante** toutes les accessibilités locales à réaliser afin de résoudre l'objectif global. On notera par ailleurs que cette définition diffère de celles introduites précédemment dans (Folschette et al., 2015b ; Paulevé, 2018) puisque les graphes que nous introduisons ici contiennent en plus les transitions des réseaux d'automates.

Définition 2.6. *Graphe de Causalité Locale (GCL)*
Étant donné un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *et une instance d'accessibilité* $\mathcal{R} = (s_0, x_i)$*, un Graphe de Causalité Locale associé à* $\mathcal{M}$ *et* $\mathcal{R}$ *est un digraphe* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ *avec* $\mathcal{V}_\mathcal{G} = \mathcal{L} \cup \mathcal{O} \cup \mathcal{Q} \cup \mathcal{T}'$ *tel que :*

- $\mathcal{L} \subset \bigcup_{y \in \Sigma} \mathcal{S}_y$ *(états locaux)*

— $\mathcal{O} \subset \bigcup_{y \in \Sigma} \mathcal{S}_y \times \mathcal{S}_y$ *(objectifs)*
— $\mathcal{Q} = \bigcup_{y_i \rightsquigarrow y_j \in \mathcal{O}} lpaths(y_i \rightsquigarrow y_j)$ *(solutions locales)*
— $\mathcal{T}' = \bigcup_{o \in \mathcal{O}} \{\eta^i \mid i \in [\![1, |\eta|]\!], \eta \in lpaths(o)\}$ *(transitions)*
— $\mathcal{E}_\mathcal{G} \subset \mathcal{V}_\mathcal{G} \times \mathcal{V}_\mathcal{G}$ *(arcs du graphe)*

Et satisfaisant les conditions suivantes :

1. $x_i \in \mathcal{L}$ *(objectif général)*
2. $(y_i, s_0(y) \rightsquigarrow y_i) \in \mathcal{E}_\mathcal{G} \quad \forall y_i \in \mathcal{L}$ *(objectifs partant de l'état initial)*
3. $(y_i \rightsquigarrow y_j, \eta) \in \mathcal{E}_\mathcal{G} \quad \forall \eta \in lpaths(y_i \rightsquigarrow y_j), \forall y_i \rightsquigarrow y_j \in \mathcal{O}$ *(toutes les solutions locales)*
4. $(y_j, y_i \rightsquigarrow y_j) \in \mathcal{E}_\mathcal{G} \quad \forall y_i, y_j \in \mathcal{L}$ *(saturation)*
5. $\forall \eta \in \mathcal{Q}, (\eta, \eta^i) \in \mathcal{E}_\mathcal{G} \quad \forall i \in [\![1, |\eta|]\!]$ *(transitions des solutions locales)*
6. $\forall \tau \in \mathcal{T}', \forall y_i \in enab(\tau), (\tau, y_i) \in \mathcal{E}_\mathcal{G}$ *(états locaux requis)*

La construction d'un Graphe de Causalité Locale se fait de manière itérative, en y ajoutant des éléments jusqu'à ce que toutes les conditions de la définition 2.6 soient satisfaites. Le premier sommet d'un GCL associé à une instance d'accessibilité est l'état local cible de l'instance (condition 1), car c'est l'état local qui doit être atteint par le réseau. Puisque cet état local doit être accessible depuis l'état initial, l'objectif partant de l'état local dans l'état initial de cet automate est également présent (condition 2). Ensuite, les différentes conditions de la définition 2.6 sont appliquées afin que le graphe contienne assez d'éléments pour représenter localement toutes les manières de résoudre l'accessibilité. Si un état local appartient au graphe, l'objectif à partir de l'état initial de cet automate jusqu'à cet état local doit être ajouté (condition 2). Cependant, il n'est pas facile de prédire quel objectif devra être utilisé dans une solution pour l'accessibilité d'un état local, puisque cela dépend des accessibilités locales précédemment résolues. C'est pourquoi tous les objectifs à partir de n'importe quel état local du graphe pour le même automate jusqu'à cet état local sont ajoutés dans le graphe (condition 4). Lorsqu'un objectif est ajouté, toutes ses solutions locales sont automatiquement ajoutées (condition 3) avec leurs transitions respectives (condition 5) et les états locaux conditions, requis pour jouer ces transitions (condition 6). Cette procédure, bien qu'elle sature rapidement le graphe, permet de n'oublier aucun élément qui serait nécessaire à la résolution du problème.

Il est également important de noter que tous les sommets n'apparaissent qu'une seule fois dans le graphe, ce qui en fait une représentation **statique** de l'exécution du modèle.

Par exemple, même si un état local $x_i \in \mathcal{S}_x$ est dans la condition de plusieurs transitions, il n'y a qu'un seul sommet $x_i \in \mathcal{V}_\mathcal{G}$ correspondant dans le graphe. Tous les sommets transitions seront donc connectés à cet unique sommet état local. Grâce à cela, la taille du graphe est polynomiale par rapport au nombre d'automates du réseau. D'un autre côté, cela induit potentiellement la présence de cycles, qui seront abordés plus tard dans ce chapitre.

Remarque 2.1. *La définition du Graphe de Causalité Locale autorise plusieurs graphes possibles pour une même instance d'accessibilité. En effet, n'importe quel graphe vérifiant l'ensemble des conditions requises est valide, et il est possible de conserver la validité en ajoutant des sommets supplémentaires dans un graphe déjà valide, bien que cela rende l'analyse de l'accessibilité plus difficile.*

Exemple 2.5. *La figure 2.3 en page 52 représente un Graphe de Causalité Locale* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ *associé à l'instance d'accessibilité* $\mathcal{R} = ((a_0, b_0, c_0), a_2)$ *pour le réseau d'automates de la figure 2.1 en page 42. Dans cette représentation, les sommets encadrés correspondent aux états locaux et aux transitions. Les objectifs sont représentés avec la notation utilisée précédemment et les sommets* $\circ$ *représentent les solutions locales (c'est-à-dire les séquences de transitions résolvant un objectif). Les nombres annotés à certains sommets sont expliqués par la suite. Dans ce GCL, les objectifs de la forme* $x_i \rightsquigarrow x_i$ *n'ont pas été ajoutés car ils n'apportent pas d'information supplémentaire sur la résolution. L'état local* b_1 *a trois prédécesseurs, car il apparaît dans la condition de trois transitions et, comme expliqué précédemment, les sommets ne sont pas dupliqués dans le graphe. Enfin, les conditions suffisantes mentionnées par la suite sont satisfaites dans ce graphe, ce qui implique que l'accessibilité de* a_2 *est possible à partir de* (a_0, b_0, c_0).

Conditions nécessaires et suffisantes pour l'accessibilité

Le Graphe de Causalité Locale a été introduit dans le but de mettre en place des conditions nécessaires et suffisantes pour la résolution de l'accessibilité dans les réseaux d'automates. Ces propriétés qui sont vérifiables rapidement garantissent en effet dans certains cas qu'une instance d'accessibilité admette une solution ou non. Sans détailler formellement ces propriétés, nous expliquons intuitivement les concepts derrière cette méthode de résolution qui seront utiles par la suite pour appréhender la nouvelle méthode que nous proposons.

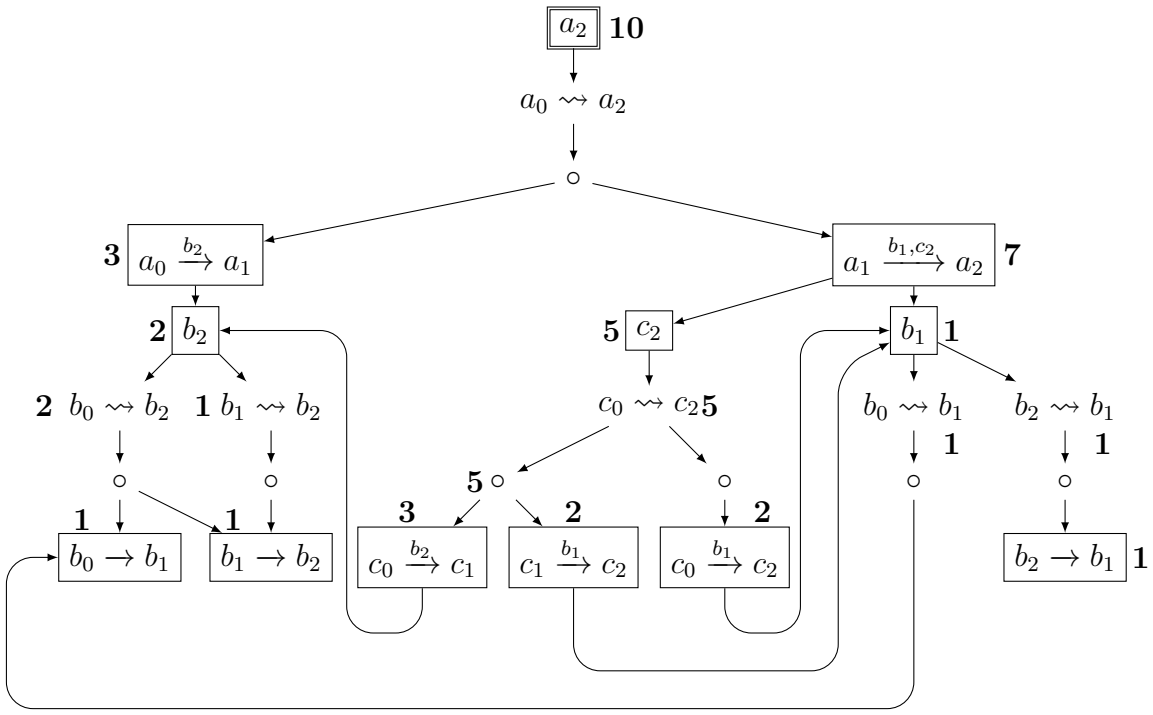

FIGURE 2.3 – Un Graphe de Causalité Locale associé à l'instance d'accessibilité $\mathcal{R} = ((a_0, b_0, c_0), a_2)$ du réseau d'automates en figure 2.1.

Les **conditions nécessaires** permettent de prouver, lorsqu'elles ne sont pas vérifiées, que l'accessibilité n'est pas possible. Elles se concentrent en particulier sur les objectifs du GCL associé à une instance d'accessibilité. Si un objectif n'admet aucune solution locale dans le GCL, on peut naturellement en déduire que cet objectif est impossible à réaliser. En propageant cette information à travers le graphe, cela permet de prouver que d'autres accessibilités locales dépendant de cet objectif ne sont pas possibles. En effet, si cet objectif est l'unique descendant d'un état local, on peut alors prouver que l'état local n'est pas accessible. En remontant encore dans le graphe, cela peut permettre de prouver qu'une transition ayant cet état local pour condition ne peut pas être jouée. Ce raisonnement local peut être propagé autant que possible dans le GCL pour prouver l'inaccessibilité de plusieurs états locaux. Cela peut permettre de simplifier un réseau d'automates pour une instance d'accessibilité donnée afin notamment d'accélérer sa vérification avec des outils exhaustifs. Enfin, si l'état local du problème d'accessibilité est prouvé non accessible, cela permet finalement de prouver que l'instance d'accessibilité n'admet pas de solutions.

De manière analogue, les **conditions suffisantes** visent à être utilisées, lorsqu'elles sont vérifiées, pour prouver qu'une accessibilité est possible. Toujours en considérant un objectif du GCL, si au moins une de ses solutions locales peut être prouvée comme étant toujours réalisable, alors cet objectif pourra être résolu dans n'importe quel cas. C'est notamment le cas si une solution locale contient des transitions n'admettant pas de conditions. Cette information peut encore une fois être propagée à travers les prédécesseurs d'un tel objectif dans un GCL. Si tous les objectifs associés à un état local peuvent être résolus, alors l'état local en question peut aussi être résolu. Enfin, si l'état local de l'instance d'accessibilité est concerné par cela, on peut en déduire que l'accessibilité admet une solution, et la solution est alors calculable via le GCL.

Nous introduisons enfin une première propriété permettant de lier le Graphe de Causalité Locale avec les solutions d'une instance d'accessibilité. La proposition suivante, qui sera réutilisée par la suite, formalise le fait que les solutions minimales, selon la notion de minimalité introduite dans la définition 2.4 en page 44, contiennent exclusivement des transitions appartenant aux GCL construits pour la même instance d'accessibilité. Cette proposition est établie dans (Paulevé, 2018) afin de simplifier un réseau d'automates en retirant des transitions qui ne sont pas utiles pour une instance d'accessibilité donnée. Sa démonstration repose sur des définitions légèrement différentes de celles des réseaux d'automates et du GCL présentées dans cette partie, puisque c'est la sémantique asynchrone généralisée (voir partie 1.2.2 du chapitre 1) qui est utilisée, et d'autres types de transitions sont également autorisés dans les modèles. La proposition reste toutefois valide ici puisque sa démonstration consiste à montrer que dans toute solution d'une instance d'accessibilité, il est possible de retirer les transitions n'appartenant pas au GCL en identifiant des boucles qui les entourent.

Proposition 2.1. *Transitions suffisantes pour l'accessibilité*
Soit $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *un réseau d'automates,* $\mathcal{R} = (s_0, x_i)$ *une instance d'accessibilité,* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ *un Graphe de Causalité Locale associé à* $\mathcal{M}$*, avec* $\mathcal{V}_\mathcal{G} = \mathcal{L} \cup \mathcal{O} \cup \mathcal{Q} \cup \mathcal{T}'$ *et* $\mathcal{R}$ *et soit* π *une solution minimale de* $\mathcal{R}$*.*
$- \forall k \in [\![1, |\pi|]\!], \pi^k \in \mathcal{V}_\mathcal{G}$ *(toutes les transitions d'une solution minimale appartiennent au Graphe de Causalité Locale)*

2.3 Méthode hybride pour résoudre l'accessibilité dans les réseaux d'automates

Comme indiqué précédemment, les instances d'accessibilité qui n'admettent pas de solutions sont difficiles à résoudre avec les outils existants, c'est-à-dire qu'il est difficile de prouver formellement qu'il n'existe aucune solution pour ces instances. Le *Bounded Model Checking* (que nous nommons BMC par la suite) est un candidat intéressant pour traiter ce cas car les solveurs logiques permettent de vérifier rapidement et de manière exhaustive des propriétés logiques sur les séquences d'exécution d'un modèle. On s'intéresse ici en particulier à l'utilisation de solveurs SAT car ceux-ci sont particulièrement performants et l'encodage SAT, qui n'a pas encore été proposé pour les réseaux d'automates, est assez naturel. Le choix de la longueur des séquences à encoder est cependant difficile pour cette approche, comme nous l'expliquons par la suite. Le schéma de résolution en BMC est représenté sur la partie supérieure de la figure 2.4. Étant donné un réseau d'automates $\mathcal{M}$ et une instance d'accessibilité $\mathcal{R}$, ainsi qu'un paramètre k déterminant la longueur des séquences à vérifier, un encodeur SAT permet de produire une expression en logique propositionnelle Φ qui est satisfaisable uniquement si $\mathcal{R}$ admet une solution. L'utilisation d'un solveur SAT permet alors de résoudre partiellement le problème. Si le solveur renvoie SAT (*satisfiable*), alors une solution est trouvée et il est prouvé que l'accessibilité est possible. Si le solveur renvoie UNSAT (*unsatisfiable*), ce qui signifie que Φ n'est pas satisfaisable, alors il est seulement possible de dire que l'accessibilité n'est pas réalisable pour des séquences de $\mathcal{M}$ de longueur inférieure ou égale à k. Dans ce cas, on ne sait pas s'il existe des solutions dont la longueur est supérieure à k.

Afin d'exploiter la méthodologie du *Bounded Model Checking* pour traiter les cas d'inaccessibilité, nous proposons de réutiliser les outils d'analyse statique introduits dans la partie 2.2.3 afin de déterminer formellement la valeur d'un paramètre k garantissant la **complétude** de la solution. Cette stratégie complétée, représentée sur la partie inférieure de la figure 2.4, consiste en la mise en place d'une propriété et d'une méthode de calcul d'un paramètre $B(\mathcal{M}, \mathcal{R})$ garantissant l'inexistence de solution dans le cas UNSAT. Intuitivement, l'utilisation d'outils dédiés aux réseaux d'automates vise à obtenir le paramètre le plus petit possible, afin de faciliter la vérification des séquences encodées en SAT. Dans cette partie, nous mettons en place l'encodage d'une instance du problème d'accessibilité en formule logique propositionnelle. Nous expliquons ensuite comment calculer le para-

mètre $B(\mathcal{M}, \mathcal{R})$ à partir du Graphe de Causalité Locale défini précédemment, et nous démontrons formellement la complétude de cette méthode de résolution.

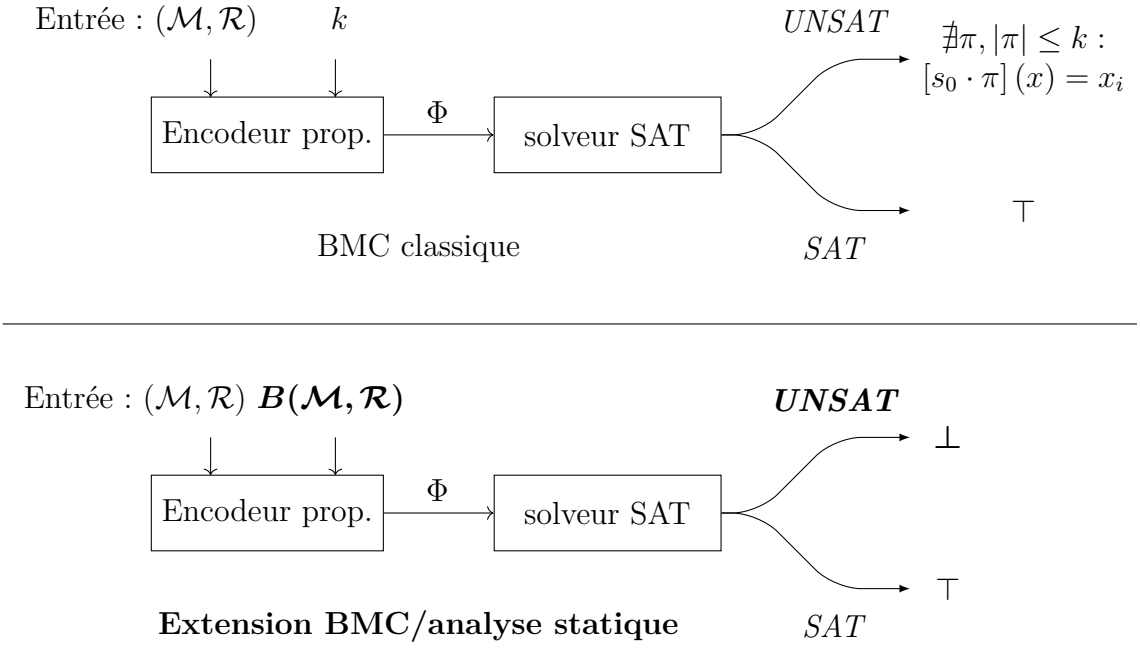

FIGURE 2.4 – Stratégies de résolution de l'accessibilité en *Bounded Model Checking* : l'approche classique avec le paramètre k est représentée en haut et l'extension avec l'analyse statique, permettant de prouver l'inaccessibilité, est représentée en bas.

2.3.1 Encodage SAT pour l'accessibilité

La première étape de notre approche de résolution consiste en la mise en place d'un encodage en logique propositionnelle de la dynamique des réseaux d'automates, afin de suivre la méthode du *Bounded Model Checking* avec les solveurs SAT. Un encodage en *Bounded Model Checking* consiste de manière générale à modéliser l'exécution d'un modèle sur un nombre fini d'itérations (k) dans un langage logique, et à utiliser des outils dédiés pour chercher une solution à la formule logique obtenue. La formule encodée vise donc à décrire l'évolution du modèle (parfois appelé dépliage du modèle) sur plusieurs itérations. Dans le cas de SAT, c'est la logique propositionnelle qui est utilisée, ce qui donne une expression logique sans quantificateurs sur des variables booléennes. L'objectif de l'encodage est donc d'obtenir une expression logique Φ_k encodant une séquence d'états $(s^1 :: s^2 :: .. :: s^{k-1} :: s^k)$ de longueur k, et telle que Φ_k est satisfaisable si et seulement s'il

existe une séquence π de transitions de taille $k-1$ résolvant une instance $\mathcal{R}$. Afin de mettre en place notre encodage, nous définissons tout d'abord les variables propositionnelles qui représentent les états d'un réseau d'automates au cours de la séquence d'exécution encodée. Ensuite, nous définissons les opérations logiques qui permettent de lier les variables propositionnelles afin de représenter les transitions possibles dans le réseau d'automate. Cette étape permet d'obtenir une expression logique représentant n'importe quelle séquence d'évolution du modèle. La dernière étape consiste enfin à ajouter des contraintes sous forme logique permettant de spécifier l'état initial et l'état local cible de l'accessibilité.

Variables propositionnelles

Afin de représenter un état $s \in \mathcal{S}$ d'un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ en logique propositionnelle, on définit un ensemble de variables propositionnelles pour chaque automate $x \in \Sigma$ de $\mathcal{M}$, avec une variable propositionnelle par état local de x, indiquant si le réseau est dans l'état local correspondant dans l'état s. Pour un état s donné, on représente l'état local d'un automate $x \in \Sigma$ de la manière suivante : $v_{s,x} = (v_{s,x_0}, v_{s,x_1}, v_{s,x_2}, ..)$ avec $v_{s,x_j} = \begin{cases} \top \text{ si } s(x) = x_j \\ \bot \text{ sinon} \end{cases}$
Cela permet de représenter s par un vecteur v_s contenant les variables propositionnelles de chaque état local de chaque automate : $v_s = (v_{s,x}, v_{s,y}, ..)$. Une séquence d'états $(s^1 :: s^2 :: .. :: s^k)$ est finalement encodée par un vecteur de variables v_{s^i} pour chaque état $i \in [\![1, k]\!]$: $(v_{s^1}, v_{s^2}, .., v_{s^k})$.

Contraintes de transition

Pour que la séquence d'états $(s^1 :: s^2 :: .. :: s^k)$ encodée représente correctement la dynamique des réseaux d'automates asynchrones, il faut s'assurer que l'expression logique obtenue ne soit satisfaisable que s'il est possible de passer d'un état s^i à son successeur s^{i+1} via la jouabilité d'une transition, pour tout $i \in [\![1, k-1]\!]$. Il est possible de modéliser cela via l'utilisation d'opérateurs logiques entre chaque groupe de variables v_{s^i} et $v_{s^{i+1}}$. On contraint ainsi la valeur de vérité d'une variable propositionnelle v_{s^{i+1},x_j}, représentant la présence de l'état local x_j de l'automate x dans l'état s^{i+1}, en vérifiant une des deux conditions suivantes : le réseau présente déjà l'état local x_j à l'itération i ($s^i(x) = x_j$), ou bien une transition menant à x_j est jouée entre l'état s^i et l'état s^{i+1}

$(\exists \tau \in \mathcal{T}_x : target(\tau) = x_j \wedge jouable(s^i, \tau) = \top)$. Dans le cas où une transition conduisant à x_j est jouée dans le modèle, il faut également s'assurer que les variables propositionnelles représentant les autres états locaux de l'automate x ne soient pas vraies. Enfin, pour vérifier la sémantique asynchrone, il faut également s'assurer que si une transition est jouée sur x pour obtenir x_j, alors les autres automates du modèle ne sont pas modifiés, ce que l'on peut formaliser par $\forall y \in \Sigma$ tel que $y \neq x : s^{i+1}(y) = s^i(y)$. Nous encodons ces vérifications dans des expressions propositionnelles $\phi_{x_j,i}$ représentant les contraintes de cohérence pour un état local x_j de l'automate x entre les états s^i et s^{i+1}, définies comme suit :

Définition 2.7. *Contraintes de transition pour l'état local x_j entre s^i et s^{i+1}*
Soit $v_{s^i}, v_{s^{i+1}}$ deux groupes de variables propositionnelles encodant deux états consécutifs s^i et s^{i+1}, et $\tau \in \mathcal{T}_x$ telle que $target(\tau) = x_j$, une transition de $\mathcal{M}$. La vérification de la jouabilité de τ pour l'obtention de x_j est encodée via l'expression :
- $\psi_{\tau,i} := v_{s^i,orig(\tau)} \wedge \left(\bigwedge_{y_j \in enab(\tau)} v_{s^i,y_j} \right) \wedge \left(\bigwedge_{x_l \neq x_j} \neg v_{s^{i+1},x_l} \right)$
L'absence de modification des états locaux d'un automate $z \neq x$ est encodée par :
- $\chi_{z,i} := \bigwedge_{z_j \in \mathcal{S}_z} (v_{s^i,z_j} \rightarrow v_{s^{i+1},z_j})$
L'encodage de la jouabilité de n'importe quelle transition pour l'obtention de x_j est donnée par :
- $\phi_{x_j,i} := v_{x_j,i+1} \rightarrow \left(\left[\left(\bigvee_{\tau_x \in \mathcal{T}_x, dest(\tau_x) = x_j} \psi_{\tau_x,i} \right) \wedge \left(\bigwedge_{z \in \Sigma, z \neq x} \chi_{z,i} \right) \right] \vee v_{s^i,x_j} \right)$

L'expression $\phi_{x_j,i}$ permet donc de s'assurer que la variable propositionnelle v_{x_j} correspondant à l'état local j de l'automate x est vraie uniquement si le modèle est autorisé à être dans l'état x_j selon sa dynamique et sa sémantique. Cette expression peut être utilisée pour chaque état local de chaque automate dans un état s^i de la séquence encodée. Cela permet de définir une expression générale représentant la transition d'un état s^i à un état s^{i+1} :

Définition 2.8. *Contraintes de transition entre s^i et s^{i+1}*
Étant donné un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ et une séquence de variables encodée v, la contrainte de transition globale entre s^i et s^{i+1} est définie par :
- $\Phi_i := \left(\bigwedge_{x_j \in \mathcal{S}_x, x \in \Sigma} \phi_{x_j,i} \right)$

L'utilisation d'expressions Φ_i pour chaque itération $i \in [\![1, k-1]\!]$ de la séquence permet alors de modéliser l'exécution complète d'une séquence de longueur k du modèle $\mathcal{M}$. On obtient enfin par conjonction logique une expression complète pour les k itérations de la séquence modélisée : $\Phi_{\mathcal{M},k} := \left(\bigwedge_{i \in [\![1,k-1]\!]} \Phi_i \right)$

Remarque 2.2. *On peut noter par ailleurs que cet encodage autorise l'absence de changement entre un état s^i et son successeur s^{i+1}, c'est-à-dire qu'une telle expression encodée peut être satisfaisable même si aucune transition du réseau n'est jouable. Bien que de telles séquences ne soient pas réellement conformes à la sémantique du modèle, une séquence valide peut facilement en être extraite en supprimant les états consécutifs qui se répètent. Cet aspect n'est pas limitant pour la résolution de l'accessibilité car cela rajoute seulement des solutions potentielles, et cela a l'avantage de permettre de trouver des séquences solutions d'une accessibilité de longueur strictement inférieure à k à partir d'un encodage de longueur k.*

Vérification de l'accessibilité

L'expression $\Phi_{\mathcal{M},k}$ définie précédemment pour un réseau d'automates $\mathcal{M}$ permet d'encoder l'exécution du réseau d'automates sur k itérations. Plus précisément, l'expression $\Phi_{\mathcal{M},k}$ est satisfaisable uniquement si la séquence d'états $(s^1 :: .. :: s^k)$ correspond à une séquence valide du modèle, ou encore à un chemin dans son graphe de transitions. Afin de modéliser une instance d'accessibilité $\mathcal{R} = (s_0, x_i)$, il faut donc rajouter un ensemble de contraintes fixant l'état initial de la séquence (s^1 représenté par v_{s^1}) à s_0, et contraindre l'état local x_i dans s^k. L'expression finale $\Phi_{\mathcal{R},k}$ pour la vérification de l'accessibilité s'obtient ainsi par conjonction en rajoutant les vérifications nécessaires :

$$\Phi_{\mathcal{R},k} := \Phi_{\mathcal{M},k} \wedge \left[\bigwedge_{y \in \Sigma} \left(\left(\bigwedge_{y_l \neq s_0(y)} \neg v_{s^1,y_l} \right) \wedge v_{s^1,s_0(y)} \right) \right] \wedge \left(\left(\bigwedge_{x_l \neq x_j} \neg v_{s^k,x_l} \right) \wedge v_{s^k,x_j} \right)$$

La formule résultante ne peut pas directement être donnée en entrée de solveurs SAT puisque ceux-ci prennent généralement en entrée une formule logique en forme normale conjonctive (c'est-à-dire une conjonction de clauses logiques, où chaque clause correspond à une disjonction). Il existe par ailleurs des approches permettant de transformer une expression en logique propositionnelle quelconque en forme normale conjonctive. Pour cet encodage, c'est l'approche présentée dans (Plaisted et al., 1986) qui a été choisie,

elle-même basée sur (Tseitin, 1983). Cette méthode consiste à introduire des variables propositionnelles intermédiaires, représentant des sous-formules qui ne s'expriment pas directement forcément sous forme de clauses. Enfin, on peut noter que l'encodage de séquence présenté ici peut être facilement réutilisé pour modéliser d'autres problèmes d'analyse. Par exemple, il est possible en faisant certaines modifications d'encoder des cycles ou encore de représenter la sémantique synchrone des réseaux d'automates.

2.3.2 Extension de l'analyse statique pour le BMC

Comme représenté en figure 2.4 en page 55, un encodage de l'accessibilité en *Bounded Model Checking* permet de résoudre le problème uniquement pour des séquences d'exécution du modèle de taille fixe. Une surestimation théorique de la longueur des séquences réalisant l'accessibilité est alors nécessaire pour prouver, dans le cas où l'instance n'admet pas de solution, que l'accessibilité n'est pas réalisable pour des séquences de longueur quelconque. On recherche ici une surestimation la plus petite possible afin de faciliter la résolution SAT pour fournir la preuve de l'inexistence. Dans cette partie, nous proposons de détourner les outils d'analyse statique présentés en partie 2.2.3 afin de définir une telle surestimation de la taille des séquences résolvant l'accessibilité. La propriété que nous introduisons dans cette partie se base sur le fait que le Graphe de Causalité Locale permet de capturer de manière suffisante toutes les manières de résoudre (localement) l'accessibilité. Ce dernier point est essentiel afin de prouver que la surestimation couvre toutes les solutions potentielles. La partie suivante définit la fonction permettant de calculer cette surestimation ainsi que la propriété formelle utilisée pour obtenir la preuve d'inaccessibilité. Ensuite, une démonstration de la propriété est proposée. Cette démonstration permet d'établir un lien plus fort entre le Graphe de Causalité Locale et les solutions d'une instance d'accessibilité que celui établi lors de l'introduction des conditions nécessaires et suffisantes. Nous détaillons également certains exemples particuliers qui compliquent l'utilisation de la propriété lorsque le Graphe de Causalité Locale contient des cycles. Nous proposons enfin plusieurs pistes d'amélioration à partir de ces cas particuliers.

Définition de la Borne de Causalité Locale

La Borne de Causalité Locale que nous introduisons ici est une fonction visant à calculer, à partir d'un Graphe de Causalité Locale, une surestimation de la longueur des solutions d'une instance d'accessibilité. Cependant, il n'est pas envisageable de suresti-

mer n'importe quelle solution car il peut en exister une infinité, et elles peuvent être arbitrairement grandes en raison de la possible présence de cycles. La Borne de Causalité Locale que nous introduisons à partir du Graphe de Causalité Locale repose donc sur la notion de **solution minimale**, introduite dans la définition 2.4 en page 44 et qui suffit à caractériser toutes les manières de résoudre une instance d'accessibilité. Le calcul de cette borne se base sur le même principe de résolution locale et récursive utilisé pour construire le GCL. La première étape consiste à surestimer la longueur des solutions locales permettant de résoudre un objectif, ce qui est obtenu à partir du nombre de transitions de la plus longue solution locale. Cependant, la résolution d'un objectif d'un point de vue global dépend également de l'accessibilité de sous-objectifs, correspondants aux conditions des transitions présentes dans les solutions locales de l'objectif initial. En supposant que ces objectifs peuvent également être surestimés, on peut ainsi définir récursivement une surestimation globale, ce qui donne la fonction suivante permettant de calculer une borne pour chaque sommet d'un GCL :

Définition 2.9. *Fonction borne B*
Soit $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *un réseau d'automates,* $\mathcal{R} = (s_0, x_i)$ *un problème d'accessibilité, et* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ *un Graphe de Causalité Locale associé à* $\mathcal{M}$ *et* $\mathcal{R}$*. Suivant la définition 2.6,* $\mathcal{V}_\mathcal{G} = \mathcal{L} \cup \mathcal{O} \cup \mathcal{Q} \cup \mathcal{T}'$ *avec* $\mathcal{L} \subset \bigcup_{y \in \Sigma} \mathcal{S}_y, \mathcal{O} \subset \bigcup_{y \in \Sigma} \mathcal{S}_y \times \mathcal{S}_y, \mathcal{Q} = \bigcup_{o \in \mathcal{O}} lpaths(o)$ *et* $\mathcal{T}' = \bigcup_{o \in \mathcal{O}} \{\eta^i \mid i \in [\![1, |\eta|]\!], \eta \in lpaths(o)\}$.
La fonction de borne associée à $\mathcal{G}$ *est la fonction* $B : \mathcal{V}_\mathcal{G} \to \mathbb{N}$ *avec :*

$$
\begin{aligned}
-B(\tau) &= 1 + \sum_{y_i \in enab(\tau)} B(y_i) && \forall \tau \in \mathcal{T}' \\
-B(\eta) &= \sum_{k \in [\![1, |\eta|]\!]} B(\eta^k) && \forall \eta \in \mathcal{Q} \\
-B(o) &= \max_{\eta \in lpaths(o)} B(\eta) && \forall o \in \mathcal{O} \\
-B(y_j) &= \max_{y_i \rightsquigarrow y_j \in \mathcal{O}} B(y_i \rightsquigarrow y_j) && \forall y_j \in \mathcal{L}
\end{aligned}
$$

La fonction de Borne de Causalité Locale permet de définir la propriété générale suivante, établissant la validité de la surestimation du nombre de transitions des solutions minimales d'un problème d'accessibilité. La démonstration de cette proposition est proposée dans la suite de cette partie.

Proposition 2.2. *Borne de causalité locale pour une instance d'accessibilité*
Soit $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ un réseau d'automates, $\mathcal{R} = (s_0, x_i)$ une instance d'accessibilité, $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ un Graphe de Causalité Locale associé à $\mathcal{M}$ et $\mathcal{R}$, et B la fonction de borne associée à $\mathcal{G}$:
*− pour toute solution **minimale** π de $\mathcal{R}$, $|\pi| \leq B(x_i)$*

L'exemple suivant illustre comment la borne peut être calculée en pratique, avec un Graphe de Causalité Locale acyclique. Le cas des Graphes de Causalité Locale cycliques sera abordé par la suite.

Exemple 2.6. *La figure 2.3 en page 52 présente les bornes calculées dans le Graphe de Causalité Locale de l'instance d'accessibilité $((a_0, b_0, c_0), a_2)$ et du Réseau d'Automates de la figure 2.1 en page 42. Pour des raisons de clarté, certaines valeurs de la fonction de borne n'ont pas été ajoutées. On peut remarquer que la borne $B(a_2) = (2+1)+(5+1+1) = 3 + 7 = 10$ surestime correctement la longueur de la solution π_2 ($|\pi_2| = 6$) donnée dans l'exemple 2.3.*

Démonstration de la borne

Nous démontrons ici la proposition 2.2. La démonstration se base sur l'identification d'un lien entre chaque solution minimale d'un problème d'accessibilité, et un Graphe de Causalité Locale construit pour cette même instance. En particulier, nous proposons une décomposition des solutions minimales en plusieurs solutions locales qui correspondent à des objectifs se trouvant dans le Graphe de Causalité Locale. Nous montrons qu'il est possible, à partir des objectifs associés à ces solutions locales, de construire une structure récursive que nous appelons arbre des objectifs et qui présente une hiérarchie similaire à celle du GCL. C'est cette structure qui permet de faire le lien entre le GCL et la longueur d'une solution minimale.

Les définitions suivantes permettent de mettre en place la décomposition de n'importe quelle solution minimale en un ensemble de solutions locales associées à des objectifs liés entre eux. La première définition vise à identifier sous forme de séquence les transitions (à partir de leurs indices) ayant une condition sur un automate donné. Les états locaux présents en condition de ces transitions sont utilisés par la suite pour former les objectifs permettant de lier une solution minimale au GCL.

Définition 2.10. *Indices des transitions de π conditionnées sur un automate y : $\mathcal{I}_{\pi,y}$*
Soit $\pi = (\pi^1 :: .. :: \pi^n)$ une solution minimale pour une instance d'accessibilité $\mathcal{R} = (s_0, x_i)$ dans un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$. Pour tout $y \in \Sigma, \mathcal{I}_{\pi,y}$ est la séquence d'indices ordonnés de toutes les transitions ayant une condition sur l'automate y :
- $enab(\pi^{\mathcal{I}^j_{\pi,y}}) \cap \mathcal{S}_y \neq \varnothing \quad \forall j \in [\![1, |\mathcal{I}_{\pi,y}|]\!]$ (toutes les transitions ont une condition sur y)
- $\mathcal{I}^j_{\pi,y} < \mathcal{I}^{j+1}_{\pi,y} \quad \forall j \in [\![1, |\mathcal{I}_{\pi,y}| - 1]\!]$ (les indices des transitions sont ordonnés)
- $enab(\pi^k) \cap \mathcal{S}_y = \varnothing \quad \forall k \notin \mathcal{I}_{\pi,y}$ (il n'y a aucune autre transition ayant une condition sur y dans la solution minimale π)

À partir des transitions identifiées avec la définition précédente, des objectifs peuvent être identifiés pour chaque automate dans la solution minimale. Les objectifs sont déterminés en mettant à la suite l'état local dans l'état initial et les états locaux qui apparaissent sur les conditions des transitions. Si deux conditions se suivent et qu'aucune transition n'est jouée sur l'automate concerné (lorsque les conditions sont identiques) alors l'objectif identifié est un objectif trivial (de la forme $x_i \rightsquigarrow x_i$). De plus, si un automate n'apparaît pas dans la solution minimale, c'est-à-dire si aucune transition ne contient une condition sur cet automate et si ce n'est pas l'automate concerné par l'accessibilité, alors aucun objectif ne sera identifié pour cet automate. Enfin, l'automate cible du problème d'accessibilité présente un objectif qui atteint l'état local de l'accessibilité.

Définition 2.11. *Objectifs sur un automate y associés à une solution minimale π : $\mathcal{O}_{\pi,y}$*
Soit π une solution minimale d'une instance d'accessibilité $\mathcal{R} = (s_0, x_i)$ dans un réseau d'automates $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$, et $\mathcal{I}_{\pi,y}$ les séquences d'indices introduites dans la définition 2.10 pour chaque automate $y \in \Sigma$.
$\mathcal{O}_{\pi,y}$ est la séquence des objectifs de l'automate $y \in \Sigma$ qui sont résolus exactement par une solution minimale dans π. Plus précisément, les états locaux de ces objectifs correspondent aux conditions sur l'automate y dans les autres transitions.
Pour chaque $z \in \Sigma \setminus \{x\}$ tel que $\mathcal{I}_{\pi,z} = \varepsilon$, on a :
- $\mathcal{O}_{\pi,z} = \varepsilon$
Pour tout $y \in \Sigma \setminus \{x\}$ tel que $\mathcal{I}_{\pi,y} \neq \varepsilon$, on a $|\mathcal{O}_{\pi,y}| = |\mathcal{I}_{\pi,y}|$ et pour x, on a $|\mathcal{O}_{\pi,x}| = |\mathcal{I}_{\pi,x}|+1$
Pour $y \in \Sigma$ tel que $\mathcal{I}_{\pi,y} \neq \varepsilon$ (incluant x), on a :
- $\mathcal{O}^1_{\pi,y} = s_0(y) \rightsquigarrow y_j$ avec $y_j \in enab(\pi^{\mathcal{I}^1_{\pi,y}}) \cap \mathcal{S}_y$
- $\mathcal{O}^i_{\pi,y} = y_j \rightsquigarrow y_k$ avec $y_j \in enab(\pi^{\mathcal{I}^{i-1}_{\pi,y}}) \cap \mathcal{S}_y$ et $y_k \in enab(\pi^{\mathcal{I}^i_{\pi,y}}) \cap \mathcal{S}_y$ si $1 < i \leq |\mathcal{I}_{\pi,y}|$

Pour x, l'automate de l'état local cible de l'accessibilité, on a de plus :

- $\mathcal{O}_{\pi,x}^{|\mathcal{I}_{\pi,x}|+1} = \begin{cases} s_0(x) \rightsquigarrow x_i \ si \ \mathcal{I}_{\pi,x} = \varepsilon \\ x_k \rightsquigarrow x_i, x_k \in enab(\pi^{\mathcal{I}_{\pi,x}^{|\mathcal{I}_{\pi,x}|}}) \cap \mathcal{S}_x \ sinon \end{cases}$

L'identification des objectifs via la définition précédente permet de décomposer la solution minimale en plusieurs ensembles de transitions, qui correspondent à des solutions locales des objectifs identifiés. Il sera montré que ces solutions locales, représentées par des ensembles de transitions, forment une partition du multiensemble des transitions de la solution minimale. Cependant, on peut noter que cette partition ne peut généralement pas être représentée par des sous-séquences consécutives, car les solutions locales peuvent être mélangées dans la solution minimale (c'est-à-dire qu'elles ne peuvent pas être complètement ordonnées dans la solution).

Définition 2.12. *Transitions associées aux objectifs : $tr(\mathcal{O}_{\pi,y})$*
Soit $\mathcal{O}_{\pi,y}$ la séquence d'objectifs identifiée à partir de la définition 2.11, associée à l'automate y dans la solution minimale π, avec $y \in \Sigma$ (incluant x). Lorsque $\mathcal{I}_{\pi,y} \neq \varepsilon$, les transitions associées à chaque objectif $\mathcal{O}_{\pi,y}^i, i \in [\![1, |\mathcal{I}_{\pi,y}|]\!]$ de $\mathcal{O}_{\pi,y}$ sont les ensembles $tr(\mathcal{O}_{\pi,y}^i)$ définis par :

$$tr(\mathcal{O}_{\pi,y}^i) = \begin{cases} \{\pi^j \mid \pi^j \in \mathcal{T}_y, j < \mathcal{I}_{\pi,y}^1\} \ si \ i = 1 \\ \{\pi^j \mid \pi^j \in \mathcal{T}_y, \mathcal{I}_{\pi,y}^{i-1} < j < \mathcal{I}_{\pi,y}^i\} \ si \ i \in [\![2, |\mathcal{I}_{\pi,y}|]\!] \end{cases}$$

et l'ensemble de transitions associé à $\mathcal{O}_{\pi,x}^{|\mathcal{O}_{\pi,x}|}$ est défini par :

$$tr(\mathcal{O}_{\pi,x}^{|\mathcal{O}_{\pi,x}|}) = \begin{cases} \{\pi^j \mid \pi^j \in \mathcal{T}_x, j > \mathcal{I}_{\pi,x}^{|\mathcal{I}_{\pi,x}|}\} & si \ \mathcal{I}_{\pi,x} \neq \varepsilon \\ \{\pi^j \mid \pi^j \in \mathcal{T}_x, j \in [\![1, |\pi|]\!]\} & sinon \end{cases}$$

Enfin, nous définissons l'arbre des objectifs à partir d'une solution minimale, visant à connecter les différents objectifs identifiés via la définition 2.11. L'arbre des objectifs lie deux objectifs entre eux si l'état cible d'un objectif (successeur) est dans la condition d'une des transitions associées à l'autre objectif (prédécesseur). Cette hiérarchie est utilisée pour lier la solution minimale au GCL, afin de démontrer que la borne calculée dans le GCL est valide.

Définition 2.13. *Arbre des objectifs associé à une solution minimale : $\mathcal{H}$*
- l'arbre des objectifs d'une solution minimale π est un digraphe $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ avec $\mathcal{V}_\mathcal{H} \subset \{\mathcal{O}_{\pi,y}^i \mid i \in [\![1, |\mathcal{O}_{\pi,y}|]\!], y \in \Sigma\}$ et $\mathcal{E}_\mathcal{H} \subset \mathcal{V}_\mathcal{H} \times \mathcal{V}_\mathcal{H}$, vérifiant $(\mathcal{O}_{\pi,y}^i, \mathcal{O}_{\pi,z}^j) \in \mathcal{E}_\mathcal{H} \iff \pi^{\mathcal{I}_{\pi,z}^j} \in tr(\mathcal{O}_{\pi,y}^i)$ (la transition ayant une condition sur z finissant l'objectif est une transition associée à l'objectif parent)

Nous définissons également un ordre partiel entre les objectifs de l'arbre, qui sera réutilisé pour la démonstration de la propriété 2.2.

Définition 2.14. *Ordre partiel dans un arbre des objectifs $\mathcal{H}$: $\prec_{\mathcal{H}}$*
Soit $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$ un arbre des objectifs construit selon la définition 2.13 à partir d'une solution minimale d'une instance d'accessibilité.
- $\prec_{\mathcal{H}} \subset \mathcal{V}_{\mathcal{H}} \times \mathcal{V}_{\mathcal{H}}$ est l'ordre partiel défini de la manière suivante : $o' \prec_{\mathcal{H}} o$ si et seulement si il existe une séquence d'objectifs $\mathcal{O} = (\mathcal{O}^1 :: \mathcal{O}^2 :: .. :: \mathcal{O}^{|\mathcal{O}|})$ telle que $o = \mathcal{O}^1, o' = \mathcal{O}^{|\mathcal{O}|}$, et $\forall i \in [\![1, |\mathcal{O}|-1]\!], (\mathcal{O}^i, \mathcal{O}^{i+1}) \in \mathcal{E}_{\mathcal{H}}$ (c'est-à-dire que o' est un successeur direct ou indirect de o dans $\mathcal{H}$).

L'exemple qui suit met en pratique les définitions précédentes et illustre la construction de l'arbre des objectifs associé à la solution minimale π_2 du problème d'accessibilité $\mathcal{R}$ présenté dans l'exemple 2.3 à la page page 45.

Exemple 2.7. *Dans cet exemple, les définitions précédentes sont appliquées à l'instance d'accessibilité $\mathcal{R} = ((a_0, b_0, c_0), a_2)$ du réseau d'automates de la figure 2.1 (page 42), avec son Graphe de Causalité Locale représenté en figure 2.3 et avec la solution minimale $\pi_2 = (b_0 \to b_1 :: b_1 \to b_2 :: a_0 \xrightarrow{b_2} a_1 :: b_2 \to b_1 :: c_0 \xrightarrow{b_1} c_2 :: a_1 \xrightarrow{b_1,c_2} a_2)$ de l'exemple 2.3.*
Les séquences d'indices pour chaque automate sont :
- $\mathcal{I}_{\pi_2,a} = \varepsilon$; $\mathcal{I}_{\pi_2,b} = (3 :: 5 :: 6)$; $\mathcal{I}_{\pi_2,c} = (6)$
Les séquences d'objectifs associées sont :
- $\mathcal{O}_{\pi_2,b} = (b_0 \rightsquigarrow b_2 :: b_2 \rightsquigarrow b_1 :: b_1 \rightsquigarrow b_1)$; $\mathcal{O}_{\pi_2,c} = (c_0 \rightsquigarrow c_2)$; $\mathcal{O}_{\pi_2,a} = (a_0 \rightsquigarrow a_2)$
Leurs transitions correspondantes sont :
- $tr(\mathcal{O}^1_{\pi_2,a}) = \{a_0 \xrightarrow{b_2} a_1, a_1 \xrightarrow{b_1,c_2} a_2\}$
- $tr(\mathcal{O}^1_{\pi_2,b}) = \{b_0 \to b_1, b_1 \to b_2\}$; $tr(\mathcal{O}^2_{\pi_2,b}) = \{b_2 \to b_1\}$; $tr(\mathcal{O}^3_{\pi_2,b}) = \varepsilon$
- $tr(\mathcal{O}^1_{\pi_2,c}) = \{c_0 \xrightarrow{b_1} c_2\}$
Ce qui mène à la construction de l'arbre des objectifs $\mathcal{H} = (\mathcal{V}_{\mathcal{H}}, \mathcal{E}_{\mathcal{H}})$ avec :
- $\mathcal{V}_{\mathcal{H}} = \{\mathcal{O}^1_{\pi_2,a}, \mathcal{O}^1_{\pi_2,b}, \mathcal{O}^2_{\pi_2,b}, \mathcal{O}^3_{\pi_2,b}, \mathcal{O}^1_{\pi_2,c}\}$
- $\mathcal{E}_{\mathcal{H}} = \{(\mathcal{O}^1_{\pi_2,a}, \mathcal{O}^1_{\pi_2,b}), (\mathcal{O}^1_{\pi_2,a}, \mathcal{O}^3_{\pi_2,b}), (\mathcal{O}^1_{\pi_2,a}, \mathcal{O}^1_{\pi_2,c}), (\mathcal{O}^1_{\pi_2,c}, \mathcal{O}^2_{\pi_2,b})\}$.
On peut par exemple justifier l'arête $(\mathcal{O}^1_{\pi_2,a}, \mathcal{O}^1_{\pi_2,b})$ par le fait que $\pi^{\mathcal{I}^1_{\pi_2,b}} = \pi^3 = a_0 \xrightarrow{b_2} a_1 \in tr(\mathcal{O}^1_{\pi_2,a})$. Selon la définition 2.14, on a par ailleurs $c_0 \rightsquigarrow c_2 \prec_{\mathcal{H}} a_0 \rightsquigarrow a_2$, ainsi que $b_2 \rightsquigarrow b_1 \prec_{\mathcal{H}} a_0 \rightsquigarrow a_2$ par exemple. L'arbre des objectifs résultant est illustré en figure 2.5, page 65. Les arêtes de l'arbre sont annotées avec les transitions contenant les conditions qui permettent de lier les objectifs entre eux.

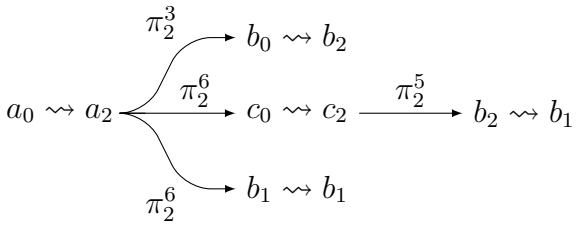

FIGURE 2.5 – Arbre des objectifs $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ de l'exemple 2.7 pour la solution minimale π_2 de l'exemple 2.3 en page 45.

L'introduction de l'arbre des objectifs pour une solution minimale d'un problème d'accessibilité nous permet finalement de lier n'importe quelle solution minimale d'une instance accessibilité à un Graphe de Causalité Locale pour la même instance. Le lemme 2.1 suivant formalise un ensemble de propriétés qui permettent d'établir ce lien. Tout d'abord, on peut remarquer qu'un ensemble de transitions $tr(o)$ d'un objectif o identifié à partir d'une solution minimale (définition 2.12) correspond à une solution locale de l'objectif o ($\in lpaths(o)$) (2.1.1). Ensuite, les ensembles de transitions identifiés à partir des objectifs o (définition 2.12) forment une partition du multiensemble [4] des transitions de la solution minimale π ($\{\!\{\pi^i \mid i \in [\![1, |\pi|]\!]\}\!\}$), ce qui indique que toutes les transitions sont capturées dans l'arbre des objectifs (2.1.2). Le troisième élément 2.1.3 met en évidence le fait que tous les objectifs identifiés sont présents dans tout Graphe de Causalité Locale associé à l'instance d'accessibilité, c'est-à-dire vérifiant la définition 2.6 (ce dernier point découle de la propriété 2.1 en page 53). Le point 2.1.4 établit que l'arbre des objectifs défini plus tôt correspond effectivement à un arbre (acyclique et connexe). Enfin le dernier point 2.1.5 indique que deux objectifs liés par une arête dans l'arbre des objectifs sont également liés dans le Graphe de Causalité Locale (via une solution locale de l'objectif prédécesseur). Ce dernier point concerne en particulier les états locaux impliqués dans les objectifs, et justifie que si le même état local est présent dans plusieurs objectifs de la solution minimale, alors il y a plusieurs arêtes correspondantes dans un LCG (ce qui est montré à partir de multiensembles).

4. Le multiensemble se justifie car plusieurs occurrences d'une même transition peuvent être présentes dans une solution minimale.

Lemme 2.1.
Soit $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ un réseau d'automates, $\mathcal{R} = (s_0, x_i)$ une instance d'accessibilité, $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ un Graphe de Causalité Locale associé à $\mathcal{M}$ et $\mathcal{R}$ et $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ l'arbre des objectifs construit à partir de π :

2.1.1 $\forall y \in \Sigma, \forall i \in [\![1, |\mathcal{O}_{\pi,y}|]\!], \exists \eta \in lpaths(\mathcal{O}^i_{\pi,y})$ *tel que*
$\{\eta^j \mid j \in [\![1, |\eta|]\!]\} = tr(\mathcal{O}^i_{\pi,y})$ *($tr(o)$ correspond à une solution locale de o).*

2.1.2 $\bigcup\limits_{o \in \mathcal{V}_\mathcal{H}} \{\!\{\tau \mid \tau \in tr(o)\}\!\} = \{\!\{\pi^i \mid i \in [\![1, |\pi|]\!]\}\!\}$ *(les multiensembles tr forment une partition, avec répétition des transitions, du multiensemble des transitions de π).*

2.1.3 $\forall y \in \Sigma, \forall i \in [\![1, |\mathcal{O}_{\pi,y}|]\!], \mathcal{O}^i_{\pi,y} \in \mathcal{V}_\mathcal{G}$ *(tous les objectifs identifiés appartiennent au Graphe de Causalité Locale).*

2.1.4 Le digraphe $\mathcal{H}$ des objectifs associé à une solution minimale est acyclique et connexe : c'est un arbre.

2.1.5 Soit $o \in \mathcal{V}_\mathcal{H}$ et soit η_o la solution locale associée à $tr(o)$ (selon le point 2.1.1 du lemme 2.1).
Alors, on a $\{\!\{y_j \mid (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_\mathcal{H}\}\!\} \subset \{\!\{y_j \mid y_j \in enab(\eta^k_o), k \in [\![1, |\eta_o|]\!]\}\!\}$ (tous les successeurs d'un objectif o dans l'arbre des objectifs sont des successeurs de la solution locale η_o de cet objectif dans le Graphe de Causalité Locale).

Démonstration. Preuve du lemme 2.1

2.1.1 Soit $y \in \Sigma, i \in [\![1, |\mathcal{O}_{\pi,y}|]\!]$. D'après la définition 2.12, page 63, π étant une solution, les transitions de $tr(\mathcal{O}_{\pi,y})$ forment un chemin local dans y. Supposons que $tr(\mathcal{O}^i_{\pi,y})$ ne corresponde pas à une solution locale, ce qui implique par la définition 2.5, page 49, qu'il existe $\pi^j, \pi^l \in tr(\mathcal{O}^i_{\pi,y}), j \neq l$, tels que $dest(\pi^j) = dest(\pi^l)$. En supprimant la boucle entre j et l dans π, la sous-séquence résultante est encore jouable car $\forall k \in [\![j+1, l-1]\!], enab(\pi^k) \cap \mathcal{S}_y = \varnothing$ (définitions 2.10 et 2.12). Cela impliquerait que π n'est pas minimale, ce qui est une contradiction.

2.1.2 D'après la définition 2.12, $\left(\bigcup\limits_{o \in \mathcal{V}_\mathcal{H}} tr(o)\right) \subset \{\!\{\pi^i \mid i \in [\![1, |\pi|]\!]\}\!\}$.
Maintenant, montrons par l'absurde que $\{\!\{\pi^i \mid i \in [\![1; |\pi|]\!]\}\!\} \subset \left(\bigcup\limits_{o \in \mathcal{V}_\mathcal{H}} tr(o)\right)$. Suppo-

sons qu'il existe $i \in [\![1, |\pi|]\!]$ avec $\pi^i \in \mathcal{T}_y$, avec $y \neq x$ et tel que $\forall k \in [\![1, |\mathcal{O}_{\pi,y}|]\!], \pi^i \notin tr(\mathcal{O}^k_{\pi,y})$. Puisque π est minimale, cela implique que $\exists j \in [\![i+1, |\pi|]\!]$ tel que $enab(\pi^j) \cap \mathcal{S}_y \neq \varnothing$ (autrement, il serait possible de retirer π^i en conservant la jouabilité de π). Supposons que j soit le plus petit indice supérieur à i tel que $enab(\pi^j) \cap \mathcal{S}_y \neq \varnothing$. Selon la définition 2.10, $enab(\pi^j) \cap \mathcal{S}_y \neq \varnothing$ implique que $\exists k \in [\![1, |\mathcal{I}_{\pi,y}|]\!]$ tel que $\mathcal{I}^k_{\pi,y} = j$. D'après la définition 2.12, page 63, ceci implique que $\pi^i \in tr(\mathcal{O}^k_{\pi,y})$, ce qui est une contradiction.
Supposons maintenant que $\pi^i \in \mathcal{T}_x$. Deux cas sont alors possibles. Dans un premier cas, si $\exists j > i$ tel que $enab(\pi^j) \cap \mathcal{S}_x \neq \varnothing$, alors comme précédemment on peut montrer que $\exists k$ tel que $\mathcal{I}^k_{\pi,x} = j$, ce qui implique $\pi^j \in tr(\mathcal{O}^k_{\pi,x})$. Dans un deuxième cas, $enab(\pi^j) \cap \mathcal{S}_x = \varnothing \quad \forall j > i \implies \pi^i \in tr(\mathcal{O}^{|\mathcal{O}_{\pi,x}|}_{\pi,x})$ d'après la définition 2.12, ce qui est une contradiction.

2.1.3 Nous traitons séparément les différents cas de la définition 2.11. Premièrement, d'après la proposition 2.1, page 53, $\forall i \in [\![1, |\pi|]\!], \pi^i \in \mathcal{V}_\mathcal{G}$. Ceci implique, d'après la définition 2.6.6 en page 49, que $\forall y_j \in enab(\pi^i) : y_j \in \mathcal{V}_\mathcal{G}$. Ainsi, d'après la définition 2.11, $\forall y \in \Sigma, \forall k \in [\![2, |\mathcal{I}_{\pi,y}|]\!], \mathcal{O}^k_{\pi,y} \in \mathcal{V}_\mathcal{G}$. Deuxièmement, puisque $y_j \in \mathcal{V}_\mathcal{G} \implies s_0(y) \rightsquigarrow y_j \in \mathcal{V}_\mathcal{G}$ (définition 2.6.2), alors $\mathcal{O}^1_{\pi,y} \in \mathcal{V}_\mathcal{G} \quad \forall y \in \Sigma$. Pour terminer, $x_i \in \mathcal{V}_\mathcal{G} \implies \mathcal{O}^{|\mathcal{O}_{\pi,x}|}_{\pi,x} \in \mathcal{V}_\mathcal{G}$.

2.1.4 D'après la définition 2.13 en page 63, et puisque chaque transition de π est associée à un seul objectif identifié (définition 2.12), tous les objectifs excepté $\mathcal{O}^{|\mathcal{O}_{\pi,y}|}_{\pi,x}$ ont exactement un prédécesseur. Ainsi, $|\mathcal{E}_\mathcal{H}| = |\mathcal{V}_\mathcal{H}| - 1$.
Montrons maintenant par l'absurde que $\mathcal{H}$ est acyclique. Supposons que $\mathcal{H}$ contienne un cycle, et soit $(\mathcal{O}^i_{\pi,x}, \mathcal{O}^{i'}_{\pi,y}), (\mathcal{O}^{i'}_{\pi,y}, \mathcal{O}^{i''}_{\pi,z}) \in \mathcal{E}_\mathcal{H}$ deux arêtes consécutives de ce cycle, impliquant $\mathcal{O}^i_{\pi,x} \prec_\mathcal{H} \mathcal{O}^{i''}_{\pi,z}$ selon la définition 2.14. D'après la définition 2.12, $\mathcal{I}^{i'}_{\pi,y} > k \quad \forall \pi^k \in tr(\mathcal{O}^{i'}_{\pi,y})$. De plus, $(\mathcal{O}^{i'}_{\pi,y}, \mathcal{O}^{i''}_{\pi,z}) \in \mathcal{E}_\mathcal{H} \implies \pi^{\mathcal{I}^{i''}_{\pi,z}} \in tr(\mathcal{O}^{i'}_{\pi,y})$ selon la définition 2.13. Ainsi, $\mathcal{I}^{i'}_{\pi,y} > \mathcal{I}^{i''}_{\pi,z}$ (les transitions d'objectifs consécutifs dans $\mathcal{H}$ sont ordonnées dans π).
De manière similaire selon la définition 2.12, $\mathcal{I}^{i''}_{\pi,z} > k \quad \forall \pi^k \in tr(\mathcal{O}^{i''}_{\pi,z})$. On peut montrer par induction, en raisonnant sur les objectifs consécutifs du cycle, que $\mathcal{I}^{i''}_{\pi,z} > k \quad \forall \pi^k \in tr(o)$ si $o \prec_\mathcal{H} \mathcal{O}^{i''}_{\pi,z}$. En particulier, puisque $\mathcal{O}^i_{\pi,x} \prec_\mathcal{H} \mathcal{O}^{i''}_{\pi,z}$, alors $\mathcal{I}^{i''}_{\pi,z} > \mathcal{I}^{i'}_{\pi,y}$. Donc, on a $\mathcal{I}^{i'}_{\pi,y} > \mathcal{I}^{i''}_{\pi,z}$ et $\mathcal{I}^{i''}_{\pi,z} > \mathcal{I}^{i'}_{\pi,y}$, ce qui est une contradiction.
Ainsi $\mathcal{H}$ est acyclique, et comme $|\mathcal{E}_\mathcal{H}| = |\mathcal{V}_\mathcal{H}| - 1$, il s'agit donc d'un arbre avec

$\mathcal{O}_{\pi,x}^{|\mathcal{O}_{\pi,x}|}$ étant sa racine.

2.1.5 Selon la définition 2.13, $\forall y_* \rightsquigarrow y_j, (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_\mathcal{H} \implies \exists k \in [\![1, |\eta_o|]\!]$ tel que $y_j \in enab(\eta_o^k)$ (pour $y_* \rightsquigarrow y_j = \mathcal{O}_{\pi,y}^i$, on a en fait $k = \mathcal{I}_{\pi,y}^i$). On a donc $\{y_j \mid (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_\mathcal{H}\} \subset \{y_j \mid y_j \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!]\}$. Montrons maintenant que les états locaux y_j répétés dans $\{\!\{y_j \mid (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_\mathcal{H}\}\!\}$ le sont aussi dans $\{\!\{y_j \mid y_j \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!]\}\!\}$.
Soit $z_l \in \mathcal{S}_z$ tel que $\exists (o, z_* \rightsquigarrow z_l) \in \mathcal{E}_\mathcal{H}$. D'après la définition 2.10, $\forall i, i' \in [\![1, |\mathcal{I}_{\pi,z}|]\!], i \neq i' : \mathcal{I}_{\pi,z}^i \neq \mathcal{I}_{\pi,z}^{i'}$. Ceci implique $|\{\!\{z_l \mid (o, z_* \rightsquigarrow z_l) \in \mathcal{E}_\mathcal{H}\}\!\}| = |\{\!\{\mathcal{I}_{\pi,z}^i \mid \pi^{\mathcal{I}_{\pi,z}^i} \in tr(o), i \in [\![1, |\mathcal{I}_{\pi,z}|]\!]\}\!\}| = |\{\mathcal{I}_{\pi,z}^i \mid \mathcal{I}_{\pi,z}^i \in tr(o), i \in [\![1, |\mathcal{I}_{\pi,z}|]\!]\}|$ (unicité des $\mathcal{I}_{\pi,z}^i$). De plus, $\pi^{\mathcal{I}_{\pi,z}^i} \in tr(o) \implies \mathcal{I}_{\pi,z}^i \in [\![1, |\eta_o|]\!] \quad \forall i \in [\![1, \mathcal{I}_{\pi,z}]\!]$. On a donc $\{\!\{z_l \mid (o, z_* \rightsquigarrow z_j) \in \mathcal{E}_\mathcal{H}\}\!\} \subset \{\!\{z_l \mid z_l \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!])\}\!\}$, ce qui prouve que les répétitions sont conservées.

□

Ce lemme permet enfin d'introduire un dernier lemme permettant de prouver que la Borne de Causalité Locale surestime formellement le nombre de transitions des solutions minimales d'une instance d'accessibilité. Le lemme 2.2 suivant établit que le nombre obtenu à partir de la fonction de la Borne de Causalité Locale appliquée à un objectif du Graphe de Causalité Locale est supérieur au nombre de transitions associées à cet objectif ainsi qu'aux objectifs descendants dans l'arbre des objectifs (en utilisant l'ordre partiel introduit dans la définition 2.14). La démonstration de ce résultat repose sur une induction structurelle sur l'arbre des objectifs. Elle utilise les différentes propriétés introduites dans le lemme 2.1 précédent, qui permettent de montrer une équivalence entre la hiérarchie de l'arbre des objectifs et celle du GCL, restreint aux seules solutions locales utilisées dans la solution minimale.

Lemme 2.2.
Soit $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ un réseau d'automates, $\mathcal{R} = (s_0, x_i)$ une instance d'accessibilité, $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ un Graphe de Causalité Locale associé à $\mathcal{M}$ et $\mathcal{R}$, π une solution minimale pour $\mathcal{R}$, et $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ l'arbre des objectifs construit à partir de π.
- $\forall o \in \mathcal{V}_\mathcal{H}, B(o) \geq |tr(o)| + \sum_{o' \prec_\mathcal{H} o} |tr(o')|$

Démonstration. Preuve du lemme 2.2
- Soit $P(o)$ la propriété : $B(o) \geq |tr(o)| + \sum_{o' \prec_{\mathcal{H}} o} |tr(o')|$, avec $o \in \mathcal{V}_{\mathcal{H}}$.
- Cas de base : soit $o \in \mathcal{V}_{\mathcal{H}}$ tel que $\nexists (o, o') \in \mathcal{E}_{\mathcal{H}}$. D'après le lemme 2.1.3, $o \in \mathcal{V}_{\mathcal{G}}$. En utilisant le lemme 2.1.1, soit $\eta_o \in lpaths(o)$ la solution locale correspondant à $tr(o)$. D'après la définition 2.9, page 60, on a $B(\eta_o) = \sum_{k \in [\![1, |\eta_o|]\!]} B(\eta_o^k) \geq \sum_{k \in [\![1, |\eta_o|]\!]} \left(1 + \sum_{y_j \in enab(\eta_o^k)} B(y_j)\right) \geq |\eta_o|$, ce qui implique que $B(o) \geq |\eta_o| \quad (= |tr(o)|)$. Ainsi, puisque o n'a pas de successeurs, $|\{o' \mid o' \prec_{\mathcal{H}} o\}| = 0$, donc $P(o)$ est valide.
- Cas inductif : soit $o \in \mathcal{V}_{\mathcal{H}}$ et supposons que $\forall o' : o' \prec_{\mathcal{H}} o, P(o')$ est valide.
De manière similaire au cas de base, le lemme 2.1.3 implique que $o \in \mathcal{V}_{\mathcal{G}}$. Soit $\eta_o \in lpaths(o)$ la solution locale qui correspond à $tr(o)$. D'après la définition 2.9,
$B(\eta_o) = \sum_{k \in [\![1, |\eta_o|]\!]} \left(\left(\sum_{y_j \in enab(\eta_o^k)} B(y_j) \right) + 1 \right) = |\eta_o| + \sum_{y_j \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!]} B(y_j)$
D'après le lemme 2.1 (point 2.1.5), $\{\!\{y_j \mid (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_{\mathcal{H}}\}\!\} \subset \{\!\{y_j \mid y_j \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!]\}\!\}$. Ceci implique $\sum_{y_j \in enab(\eta_o^k), k \in [\![1, |\eta_o|]\!]} B(y_j) \geq \sum_{y_j : (o, y_* \rightsquigarrow y_j) \in \mathcal{E}_{\mathcal{H}}} B(y_j)$.
De plus, d'après la définition 2.9 :
- $B(o) \geq B(\eta_o)$ et $B(y_j) \geq B(y_* \rightsquigarrow y_j) \quad \forall y_* \rightsquigarrow y_j \in \mathcal{V}_{\mathcal{G}}, \forall y \in \Sigma$.
Ce qui conduit à $B(o) \geq |\eta_o| + \sum_{o' : (o, o') \in \mathcal{E}_{\mathcal{H}}} B(o')$. Ensuite P peut être appliqué inductivement à tous les objectifs descendants de o dans $\mathcal{H}$. Ce qui mène à :
$B(o) \geq |\eta_o| + \sum_{o' : (o, o') \in \mathcal{E}_{\mathcal{H}}} \left(|tr(o')| + \sum_{o'' \prec_{\mathcal{H}} o'} |tr(o'')| \right)$.
$\mathcal{H}$ étant un arbre, on a :
$\{\!\{o' \mid o' \prec_{\mathcal{H}} o\}\!\} = \bigcup_{o' : (o, o') \in \mathcal{E}_{\mathcal{G}}} \left(\{o'\} \cup \{\!\{o'' : o'' \prec_{\mathcal{H}} o'\}\!\} \right)$, ce qui implique $B(o) \geq |\eta_o| + \sum_{o' : o' \prec_{\mathcal{H}} o} |tr(o')|$. Puisque $|\eta_o| = |tr(o)|$, on en conclut que $P(o)$ est également valide. □

Ce dernier lemme, quand il est appliqué à la racine de l'arbre (c'est-à-dire $\mathcal{O}_{\pi,x}^{|\mathcal{O}_{\pi,x}|}$), prouve que la fonction de borne, appliquée à x_i dans le Graphe de Causalité Locale permet de surestimer la longueur de la solution minimale π. En effet, le lemme 2.1.2 implique que $\sum_{o \in \mathcal{V}_{\mathcal{G}}} |tr(o)| = |\pi|$. De plus, puisque tous les objectifs identifiés à partir de la solution minimale appartiennent au Graphe de Causalité Locale, la racine de l'arbre des objectifs $\mathcal{H}$ est un successeur de x_i dans le GCL. Ainsi, $B(x_i) \geq B(\mathcal{O}_{\pi,x}^{|\mathcal{O}_{\pi,x}|})$ permet de correctement surestimer $|\pi|$.

Exemple 2.8. *En reprenant l'exemple 2.6, il est possible de vérifier que l'association entre le GCL de la figure 2.3 en page 52 et l'arbre des objectifs représenté en figure 2.5 en page 65 est possible. En particulier, il est possible de vérifier que les propriétés du lemme 2.1 sont vérifiées. Par exemple, la propriété 2.1.5 est vérifiée pour les objectifs* $a_0 \rightsquigarrow a_2$ *et* $c_0 \rightsquigarrow c_2$ *qui sont connectés dans l'arbre des objectifs, et également dans le GCL par l'intermédiaire de la transition* $a_1 \xrightarrow{b_1,c_2} a_2$*, qui correspond à* π_2^6 *dans la solution donnée dans l'exemple 2.3 en page 45.*

2.3.3 Mise en pratique et implémentation

Dans cette partie, nous nous concentrons sur la mise en pratique de notre méthode de résolution hybride pour le problème d'accessibilité. Dans un premier temps, nous abordons certains cas difficiles pour lesquels la résolution nécessite une analyse approfondie sur le Graphe de Causalité Locale. Nous donnons des pistes pour surmonter les difficultés pouvant être rencontrées. Enfin, nous proposons une implémentation complète permettant de tester en pratique le calcul de la borne de causalité locale ainsi que l'encodage SAT.

Application de la borne dans le cas cyclique

La proposition 2.2 (page 61) peut s'appliquer à n'importe quelle instance de problème d'accessibilité à partir du moment où la fonction de borne (définition 2.9, page 60) est calculable. Cependant, lorsqu'un objectif du Graphe de Causalité Locale appartient à un cycle, sa borne ne peut pas être calculée car elle dépend indirectement d'elle-même. Intuitivement, on peut penser que la présence d'un cycle dans un GCL rend l'accessibilité impossible. C'est en effet le cas lorsque deux automates dépendent l'un de l'autre pour réaliser leur accessibilité locale, et qu'aucune autre solution n'est disponible. La suppression de ce type de cycle dans un GCL n'invalide donc pas la proposition 2.2, et le calcul des bornes des sommets n'apparaissant pas dans le cycle suffit à surestimer la longueur des solutions minimales. Nous illustrons ceci dans l'exemple 2.9.

Exemple 2.9. *Soit* $\mathcal{M}$ *le réseau d'automates représenté en figure 2.6,* $\mathcal{R} = ((a_0, b_0), a_1)$ *une instance d'accessibilité et* $\mathcal{G}$ *le Graphe de Causalité Locale associé à* $\mathcal{R}$*, également représenté en figure 2.6.*
Dans cet exemple, $\mathcal{R}$ *n'admet aucune solution. Plus précisément,* a_1 *ne peut pas être obtenu à partir de* a_0 *puisque la seule transition locale réalisant ce changement nécessite*

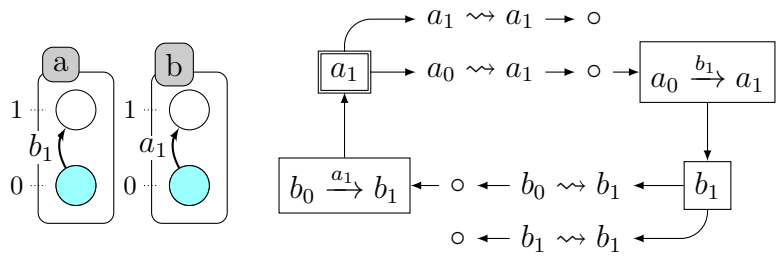

FIGURE 2.6 – Représentation graphique du réseau d'automates et du Graphe de Causalité Locale associé à l'instance $\mathcal{R} = ((a_0, b_0), a_1)$ de l'exemple 2.9.

b_1, qui nécessite a_1 (afin d'être atteint à partir de b_0). Cette dépendance mutuelle crée le cycle suivant dans le Graphe de Causalité Locale associé : $(a_1, a_0 \rightsquigarrow a_1, \circ_{a_0 \rightsquigarrow a_1}, a_0 \xrightarrow{b_1} a_1, b_1, b_0 \rightsquigarrow b_1, \circ_{b_0 \rightsquigarrow b_1}, b_0 \xrightarrow{a_1} b_1)$*. Ainsi, les bornes de causalité locale de tous ces sommets ne sont pas calculables, en particulier celle de l'objectif d'accessibilité : a_1. Cependant, l'absence de solution dans ce cas autorise la suppression de certains sommets du cycle. Puisque $a_0 \rightsquigarrow a_1$ et $b_0 \rightsquigarrow b_1$ sont tous deux dans le cycle, il est possible de supposer que a_1 ne peut pas être atteint à partir de (a_0, b_0), et de manière similaire b_1 ne peut pas être atteint à partir de (a_0, b_0). Ainsi, il est possible de retirer sans risque la solution locale $\circ_{b_0 \rightsquigarrow b_1}$ par rapport à $\mathcal{R}$ pour ce Graphe de Causalité Locale. Cela donne un sous graphe $\mathcal{G}'$ où la borne de causalité locale de a_1 peut être calculée, et est égale à 1. Puisqu'il n'existe pas de solution pour $\mathcal{R}$, 1 est en effet une surestimation correcte. Par ailleurs, il est important de remarquer que la borne calculée à partir du sous-graphe $\mathcal{G}'$ est uniquement valide pour $\mathcal{R}$. En effet, dans $\mathcal{G}'$, la borne de b_1 est égale à 0 mais b_1 peut être obtenu en une transition à partir de (a_1, b_0). Puisque l'objectif $a_1 \rightsquigarrow a_1$ est présent dans $\mathcal{G}$, la proposition 2.2 est donc valide pour $(b_1, (a_1, b_0))$ dans $\mathcal{G}$ mais elle ne l'est pas dans $\mathcal{G}'$.*

La mise en pratique de la suppression de cycles proposée dans l'exemple 2.9 peut se faire via la mise en place de propriétés sur les arbres des objectifs. En effet, ceux-ci peuvent toujours être construits à partir de solutions minimales, mais un arbre quelconque ne correspond pas forcément à un arbre des objectifs. Il serait donc intéressant d'identifier certaines propriétés sur les arbres des objectifs permettant d'exclure des sommets de Graphes de Causalité Locale. D'autres types de propriétés peuvent également être envisagées à partir des conditions nécessaires pour l'accessibilité établies sur les réseaux d'automates présentées dans la partie 2.2.3

Dans le cas général cependant, il n'est pas toujours possible de supprimer des arêtes du GCL et de conserver la propriété de la borne. En effet, certaines solutions minimales d'une instance d'accessibilité contiennent plusieurs fois la même transition. Si cette transition fait partie d'un cycle, cela signifie que différents parcours du cycle sont effectués dans la solution minimale, et que la suppression de n'importe quelle arête du cycle empêche le calcul d'une borne valide. L'exemple 2.10 ci-dessous illustre ce cas avec une instance d'accessibilité dont l'unique solution minimale contient deux transitions en double, appartenant à un cycle dans le GCL.

Exemple 2.10. *Soit* $\mathcal{M} = (\Sigma, \mathcal{S}, \mathcal{T})$ *le réseau d'automates représenté en figure 2.7 et* $\mathcal{R} = (s_0, a_1)$ *avec* $s_0 = (a_0, b_0, c_0, d_0)$. *Soit* $\mathcal{G} = (\mathcal{V}_\mathcal{G}, \mathcal{E}_\mathcal{G})$ *un GCL associé à* $\mathcal{M}$ *et* $\mathcal{R}$, *représenté en figure 2.8. La solution* $\pi = (c_0 \xrightarrow{d_0} c_1 :: d_0 \xrightarrow{c_1} d_1 :: c_1 \rightarrow c_0 :: b_0 \xrightarrow{c_0,d_1} b_1 :: d_1 \xrightarrow{b_1} d_0 :: c_0 \xrightarrow{d_0} c_1 :: d_0 \xrightarrow{c_1} d_1 :: a_0 \xrightarrow{b_1,c_1,d_1} a_1)$ *est une solution minimale de* $\mathcal{R}$.
Cette solution contient deux transitions dupliquées : $c_0 \xrightarrow{d_0} c_1$ *et* $d_0 \xrightarrow{c_1} d_1$. *Ces transitions sont également impliquées dans un cycle dans* $\mathcal{G}$.
L'association entre π *et* $\mathcal{G}$ *décrite en sous-section 2.3.2 peut être mise en place par les séquences d'indices de transitions et d'objectifs suivantes :*
$\mathcal{I}_{\pi,a} = \varepsilon; \mathcal{I}_{\pi,b} = (5 :: 8); \mathcal{I}_{\pi,c} = (2 :: 4 :: 7 :: 8); \mathcal{I}_{\pi,d} = (1 :: 4 :: 6 :: 8)$
$\mathcal{O}_{\pi,a} = (a_0 \rightsquigarrow a_1); \mathcal{O}_{\pi,b} = (b_0 \rightsquigarrow b_1 :: b_1 \rightsquigarrow b_1);$
$\mathcal{O}_{\pi,c} = (c_0 \rightsquigarrow c_1 :: c_1 \rightsquigarrow c_0 :: c_0 \rightsquigarrow c_1 :: c_1 \rightsquigarrow c_1);$
$\mathcal{O}_{\pi,d} = (d_0 \rightsquigarrow d_0 :: d_0 \rightsquigarrow d_1 :: d_1 \rightsquigarrow d_0 :: d_0 \rightsquigarrow d_1)$
Ce qui mène à l'arbre des objectifs $\mathcal{H} = (\mathcal{V}_\mathcal{H}, \mathcal{E}_\mathcal{H})$ *représenté en figure 2.9, avec :*
$\mathcal{E}_\mathcal{H} = \{(\mathcal{O}^1_{\pi,a}, \mathcal{O}^2_{\pi,b}), (\mathcal{O}^1_{\pi,a}, \mathcal{O}^4_{\pi,c}), (\mathcal{O}^1_{\pi,a}, \mathcal{O}^4_{\pi,d}), (\mathcal{O}^4_{\pi,d}, \mathcal{O}^3_{\pi,c}), (\mathcal{O}^3_{\pi,c}, \mathcal{O}^3_{\pi,d}), (\mathcal{O}^3_{\pi,d}, \mathcal{O}^1_{\pi,b}),$
$(\mathcal{O}^1_{\pi,b}, \mathcal{O}^2_{\pi,c}), (\mathcal{O}^1_{\pi,b}, \mathcal{O}^2_{\pi,d}), (\mathcal{O}^2_{\pi,d}, \mathcal{O}^1_{\pi,c}), (\mathcal{O}^1_{\pi,c}, \mathcal{O}^1_{\pi,d})\}.$
Concentrons-nous sur la borne de c_1. *En supposant qu'il est possible de couper le cycle dans* $\mathcal{G}$, *on retirerait alors l'arête* $(d_0 \xrightarrow{c_1} d_1, c_1)$, *ce qui mène à une borne de* 0 *pour* d_1 *(seules les solutions triviales restent valides). De ce fait, la borne de* c_1 *serait égale à* 4. *Cependant, selon la Borne de Causalité Locale, on devrait avoir* $B(c_0 \rightsquigarrow c_1) \geq |tr(c_0 \rightsquigarrow c_1)| + \sum_{o \prec_\mathcal{H} c_0 \rightsquigarrow c_1} |tr(o)|$, *plus précisément* $B(c_0 \rightsquigarrow c_1) \geq |tr(c_0 \rightsquigarrow c_1)| + |tr(d_1 \rightsquigarrow d_0)| + |tr(b_0 \rightsquigarrow b_1)| + |tr(c_1 \rightsquigarrow c_0)| + |tr(d_0 \rightsquigarrow d_1)| + |tr(c_0 \rightsquigarrow c_1)| + |tr(d_0 \rightsquigarrow d_0)| \implies B(c_0 \rightsquigarrow c_1) \geq \mathbf{6} > 4.$
On peut également vérifier ceci sur l'arbre des objectifs de π *en figure 2.9 puisque l'objectif* $c_0 \rightsquigarrow c_1$ *est descendant de lui-même, ce qui signifie l'obligation d'utiliser certains objectifs deux fois pour la solution* π. *L'association entre le Graphe de Causalité Locale et la solution minimale n'est donc pas possible dans le cas présent si le cycle est coupé, car cela invalide la proposition 2.2.*

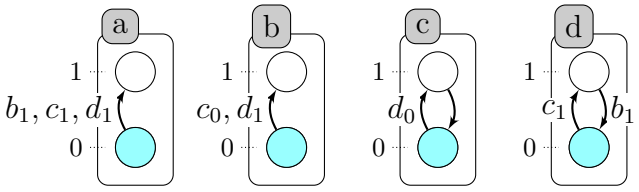

FIGURE 2.7 – Réseau d'automates de l'exemple 2.10 .

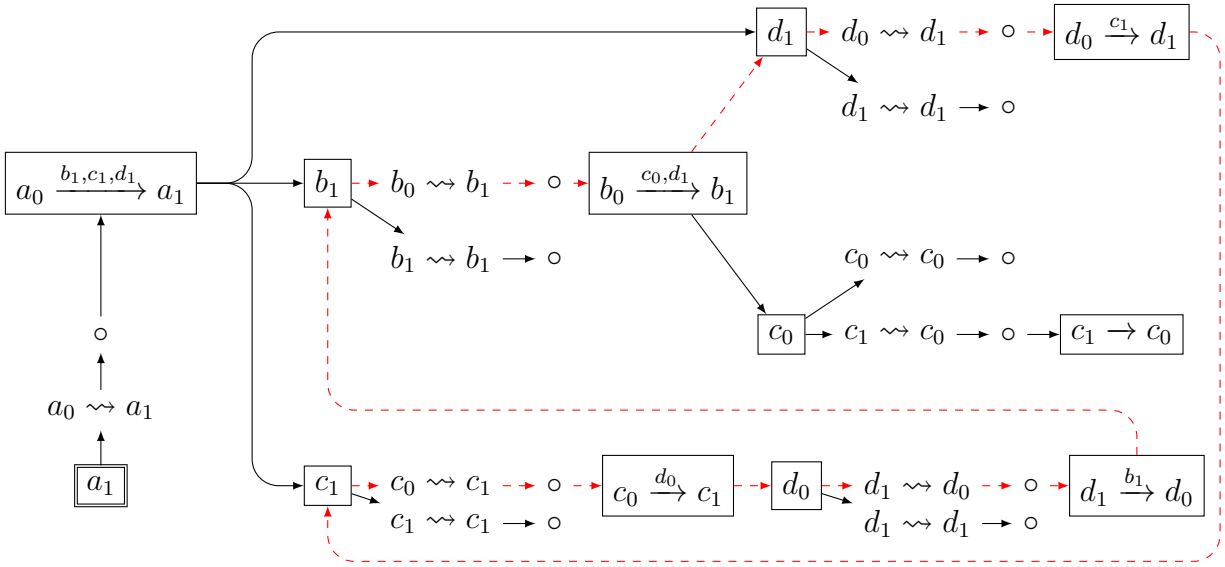

FIGURE 2.8 – Graphe de Causalité Locale associé à l'exemple 2.10, contenant un cycle.

L'exemple 2.10 illustre donc un cas particulier dans lequel la seule solution minimale d'une instance d'accessibilité contient plusieurs fois certaines transitions, formant un cycle irréductible dans le GCL associé. Ces cas semblent à première vue non surmontables pour l'application générale du calcul de la Borne de Causalité Locale, mais des raffinements du GCL sont possibles pour y remédier. En effet, une des difficultés dans cet exemple vient de l'approximation locale induite par le GCL, qui représente la dynamique du réseau de manière compacte. Nous suggérons qu'il est alors possible de raffiner le Graphe de Causalité Locale via la duplication de certains sommets du graphe. Une telle duplication constituerait en effet un dépliage du graphe, permettant de faire apparaître certaines structures plus simples. L'utilisation de propriétés supplémentaires pourrait alors permettre de supprimer certains sommets, comme nous l'avons suggéré précédemment. On peut enfin conjecturer que l'exemple 2.10 est un cas rare et qu'en pratique, les solutions minimales ne présentent pas ce type de combinatoire (ce qui semble se confirmer dans

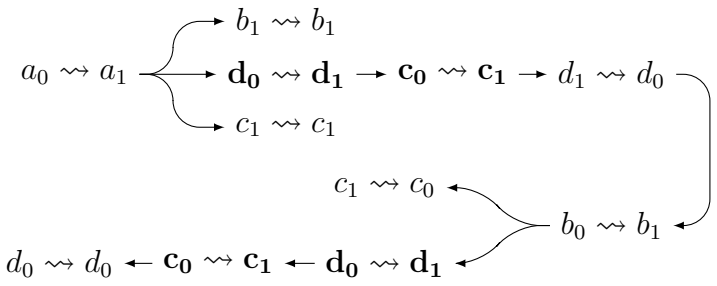

FIGURE 2.9 – L'arbre des objectifs correspondant à la solution minimale π de l'exemple 2.10.

des cas réels dans (PAULEVÉ et al., 2012)). Dans un tel cas, les stratégies mentionnées précédemment pourraient être particulièrement avantageuses.

Implémentation de la méthode

Nous avons implémenté notre méthode de résolution de l'accessibilité présentée ici dans le langage C++ afin de concrétiser sa mise en pratique. Les modèles utilisés dans cette implémentation ont été repris de ceux fournis avec le programme *Pint* [5] implémentant l'analyse statique des réseaux d'automates. Notre implémentation est disponible à l'adresse suivante : https://github.com/smbct/AAN-reach.git. Elle consiste en une première partie sur l'encodage SAT, et une seconde partie sur la borne de causalité locale. La partie encodage SAT permet de créer automatiquement une instance SAT au format DIMACS [6], et peut s'interfacer avec plusieurs solveurs SAT tels que MiniSat (EÉN et al., 2004) ou encore glucose (AUDEMARD et al., 2014). L'instance SAT peut notamment encoder la vérification de l'accessibilité, mais d'autres vérifications sont également possibles comme l'application de la technique de *k-induction* mentionnée dans la partie 2.2.2 (page 45). La seconde partie, sur l'analyse statique, implémente la construction d'un Graphe de Causalité Locale correspondant à une instance d'accessibilité. Lorsque le GCL est acyclique, il est possible de calculer la Borne de Causalité Locale et de l'utiliser dans l'encodage SAT.

Plusieurs instances d'accessibilité sont proposées à titre d'exemple dans cette implémentation, à partir de réseaux booléens aléatoires ainsi que de modèles jouets. Ces exemples, que l'on peut retrouver dans le fichier *README.md* permettent de comparer

5. https://github.com/pauleve/pint.git
6. https://www.cs.utexas.edu/users/moore/acl2/manuals/current/manual/index-seo.php/SATLINK_____DIMACS

de manière informelle le calcul de la borne de causalité locale avec celui de la borne de *k-induction*, qui résulte généralement en une borne plus grande. Certains exemples illustrent également des cas où la borne de causalité locale ne peut pas être calculée à cause de la présence de cycles dans le GCL associé. Enfin, nous proposons des instances pour lesquelles les méthodes d'analyse statique ne permettent pas de conclure (les conditions suffisantes sont vérifiées et les conditions nécessaires ne le sont pas) mais pour lesquelles la borne de causalité locale est tout de même calculable.

2.4 Discussions

Nous avons vu dans ce chapitre que le thème de l'analyse des modèles qualitatifs en biologie des systèmes conduit à des problèmes algorithmiques difficiles à résoudre. Bien que des méthodes d'exploration naïves permettent d'énumérer les solutions de ces problèmes, elles sont inefficaces sur des modèles de grande taille. Cependant, la réutilisation de méthodes de résolution développées dans différents champs de recherche en informatique, comme le *Bounded Model Checking*, permet la mise en place rapide d'approches de résolution efficaces. Il subsiste toutefois des cas difficiles à traiter, comme la démonstration de l'inaccessibilité dans les réseaux d'automates. L'approche de résolution présentée ici témoigne qu'en étudiant intensivement un problème spécifique, et notamment en combinant plusieurs approches de résolution existantes, on peut améliorer le pouvoir de résolution sur ce problème. Il s'agit d'un aspect encourageant pour la recherche sur l'analyse formelle des modèles biologiques.

Dans le cas de la biologie des systèmes, plusieurs problèmes transverses se présentent en addition à la difficulté algorithmique des problèmes rencontrés. En effet, s'agissant de la modélisation d'un système réel, il est également important de s'assurer que la formalisation du problème peut être associée à une question pertinente sur le comportement du système modélisé. Dans le cas du problème d'accessibilité, et particulièrement dans les modèles des systèmes biologiques, on peut par exemple se demander s'il est indispensable de se concentrer sur sa résolution théorique. Par exemple, si une instance d'accessibilité n'admet qu'une unique solution, et que la longueur de cette solution est particulièrement grande, on peut se demander si cela correspond à un comportement vraisemblable du système réel, et s'il est alors utile de le connaître. Dans le cas contraire, prouver informellement qu'une accessibilité est difficile à réaliser, sous-entendu que les solutions la réalisant sont longues,

peut se faire avec une approche en *Bounded Model Checking* classique. Pour l'étude d'un système biologique, il pourrait être intéressant d'appliquer une approche parcimonieuse, visant la capture des comportements qui ont le plus de chance de se produire. Il s'agit d'un principe notamment utilisé dans d'autres domaines de la bio-informatique comme en phylogénie (Camin et al., 1965). En biologie des systèmes, des modélisations discrètes probabilistes et stochastiques comme proposées dans (Kauffman, 1969 ; Shmulevich et al., 2002) permettent de considérer cet aspect, et des méthodes d'analyse probabilistes ont notamment été développées pour ce type de modèle (Heath et al., 2008 ; Katoen, 2016).

L'analyse de modèles discrets repose donc sur de nombreux outils, permettant de traiter un large spectre de formalismes y compris sur des modèles de grande taille. Cependant, les modèles qualitatifs utilisés dans la recherche sont en majorité des modèles inférés en combinant manuellement des observations expérimentales avec des connaissances de la littérature scientifique. Devant le potentiel des méthodes d'analyse de réseaux, on peut alors se concentrer sur les approches automatiques permettant d'obtenir ce type de modèle. En effet, les méthodes automatiques ont l'avantage de produire très rapidement des modèles de grande taille, et décrivant ainsi le système de manière plus précise et plus complète. L'amélioration des technologies expérimentales permet d'obtenir de grands volumes de données biologiques, ce qui nous encourage davantage à nous concentrer sur les méthodes d'inférence de modèles. Le chapitre suivant concerne cette question, en considérant en particulier l'utilisation de données d'expression génétique pour la modélisation qualitative.

Chapitre 3

Apprentissage logique des modèles qualitatifs

Résumé

Ce chapitre aborde la question de l'inférence de modèles qualitatifs d'un système biologique. Nous nous intéressons en particulier aux approches d'inférence automatique, reposant essentiellement sur les données expérimentales telles que les données d'expression génétique. Nous passons en revue les différentes approches proposées dans ce domaine. Nous nous concentrons ensuite sur l'inférence de programmes logiques, et en particulier sur le framework *LFIT* pour l'apprentissage de la dynamique d'un système. Nous discutons des limites de *LFIT* et nous en proposons enfin une extension basée sur une méthode d'optimisation afin de le rendre plus robuste aux données expérimentales. Notre extension du framework *LFIT*, appelée *Learning Optimized Logical Hypothesis* (*LOLH*), a été inspirée par les données de séquençage *single-cell*. L'algorithme *LOLH* est illustré dans ce chapitre sur un exemple artificiel, et sera appliqué à un jeu de données *single-cell* dans le chapitre 4.

3.1 Méthodes d'inférence des modèles qualitatifs

L'inférence des modèles qualitatifs est le domaine qui consiste à développer des méthodes permettant de calculer des modèles qualitatifs pour un système biologique donné. Ce problème émerge à partir du constat que l'élaboration manuelle de modèles qualitatifs est un processus chronophage. Le développement de méthodes automatiques est

également fortement encouragé par l'accumulation de grandes quantités de données via, entre autres, les technologies de séquençage. Plusieurs approches automatiques ont ainsi été développées pour répondre à ce problème. Ces méthodes sont basées sur des sources de données et des hypothèses variées. Nous détaillons dans la sous-partie suivante les différents objectifs concernant l'inférence des modèles qualitatifs, ainsi que les obstacles pouvant se présenter. Nous passons ensuite en revue différentes approches proposées dans la littérature.

3.1.1 Objectifs généraux

Il est important dans un premier temps de définir les objectifs et les attentes concernant les modèles qualitatifs. De manière générale un modèle vise à être utilisé pour améliorer la compréhension d'un système réel. Dans le cas de la modélisation qualitative des réseaux biologiques, on s'attend à ce que le modèle inféré nous apporte de nouvelles connaissances dynamiques sur le système, à savoir des comportements non connus/non observés et des relations entre variables non suspectées, qui seraient révélées via ses propriétés dynamiques. Une difficulté majeure consiste donc à déterminer rigoureusement si un modèle inféré automatiquement à partir de données est cohérent par rapport au système biologique étudié. En effet, il n'est pas suffisant de garantir que le modèle est cohérent par rapport aux données à partir desquelles il est inféré, car celui-ci peut l'être sans pour autant reproduire des comportements observables sur le système réel. Il est possible de faire la parallèle entre cet aspect, et le phénomène d'*overfitting* en apprentissage automatique. L'*overfitting* d'un modèle désigne le fait que ce modèle détient une bonne capacité prédictive sur les données sur lesquelles il est inféré, mais que ses capacités ne se généralisent pas sur tout autre donnée. Par exemple, il est possible de tester cela en pratique en excluant volontairement une partie des données lors de l'inférence du modèle (méthode de la validation croisée), ou bien en testant de nouveaux comportements expérimentalement sur le système réel. Il convient donc de définir des méthodologies robustes pour l'évaluation de méthodes d'inférence, en particulier dans le cas des modèles qualitatifs qui représentent des comportements complexes de manière abstraite. La forme de validation la plus forte semble alors être la validation par l'expérimentation. Ce type de validation n'est cependant pas systématiquement utilisable car les expérimentations sont souvent coûteuses et peu accessibles aux concepteurs de méthodes d'inférence.

3.1.2 Méthodes pour l'inférence automatique

L'approche manuelle a dans un premier temps été utilisée pour l'inférence de modèles qualitatifs dynamiques. Les données utilisées pour l'inférence manuelle sont généralement des connaissances abstraites sur le système étudié, obtenues de manière expérimentale ou à partir de la littérature. Par exemple, l'approche manuelle a été utilisée dans (Gonzalez et al., 2008) pour la mise en place d'un réseau booléen modélisant le développement de la drosophile. Dans ce type d'application, les connaissances extraites de la littérature consistent alors en des relations directes entre les variables du réseau, pouvant être directement utilisées pour déterminer les fonctions booléennes du modèle. La nécessité de développer des méthodes automatiques apparaît alors pour deux raisons différentes. Une première raison provient de la difficulté d'intégrer des propriétés dynamiques complexes de manière manuelle dans un modèle. Par exemple, il peut être intéressant d'inclure dans l'inférence d'un modèle des propriétés dynamiques telles que des accessibilités, des points fixes, ou encore des comportements dynamiques décrits en logique temporelle. La prise en compte de ces propriétés nécessite intuitivement de tester plusieurs modèles candidats et de vérifier pour chacun d'entre eux si elles sont respectées, ce qui n'est pas réalisable à la main même pour des réseaux de taille modérée. La deuxième raison motivant la mise en place de méthodes automatiques provient de l'intention d'inférer des modèles de grande taille à partir de données en grand nombre. Par exemple, comme nous l'avons mentionné dans le chapitre 1, plusieurs bases de données proposent des réseaux d'interaction moléculaire pour différents systèmes biologiques. Les données expérimentales, bien que souvent bruitées, constituent également une source de données très riche.

On peut distinguer deux stratégies différentes pour l'inférence automatique des modèles qualitatifs. Une première stratégie consiste à énumérer l'ensemble de tous les modèles cohérents avec des connaissances (graphe d'interaction, propriété dynamique, etc.) fournies en entrée. Cette stratégie est par exemple utilisée dans (Khalis et al., 2009) pour l'inférence de réseaux de Thomas à partir notamment de propriétés connues sur le système, exprimées en logique temporelle. Ces derniers travaux ont notamment donné lieu au programme d'inférence TotemBioNet [1], qui utilise l'outil de vérification formelle NuSMV [2]. Une approche analogue est employée dans (Corblin et al., 2010), pour l'inférence de modèle de Thomas via l'utilisation d'un formalisme de contraintes, et effectuant

1. TotemBioNet est disponible à l'adresse : https://gitlab.com/totembionet/totembionet.
2. NuSMV est disponible à l'adresse : https://nusmv.fbk.eu/NuSMV/.

une résolution à partir de solveurs *SAT*. Cette approche a notamment été étendue dans (Corblin et al., 2012), où des données expérimentales sont également utilisées, et où la modélisation en *ASP* est utilisée pour calculer les modèles consistants. Enfin, cette stratégie d'inférence est employée dans (Chevalier et al., 2019) pour la synthèse de réseaux booléens à partir d'un graphe d'influence et de propriétés dynamiques sur le système modélisé. Cette dernière approche repose également sur l'utilisation du formalisme *ASP*, et permet ainsi d'inférer des réseaux booléens contenant 200 composants.

La deuxième stratégie consiste à formuler l'inférence d'un modèle unique sous la forme d'un problème d'optimisation, et à rechercher le modèle obtenant le meilleur score possible pour ce problème. La résolution de ce problème d'optimisation est souvent réalisée de manière approchée, via l'utilisation d'algorithmes heuristiques. Par exemple, l'inférence de réseaux booléens est proposée dans (Barman et al., 2018) avec l'utilisation d'un algorithme génétique. Dans (Lim et al., 2016), un algorithme de recherche locale avec population est utilisé pour optimiser des réseaux booléens asynchrones à partir de données de séquençage *single-cell*. Cette deuxième stratégie vise particulièrement l'inférence de modèle à partir de données expérimentales, comme nous l'abordons par la suite. De manière intéressante, devant les nombreuses méthodes proposées pour l'inférence des réseaux de régulation, des efforts de synthèses et des démarches participatives ont été développés afin de caractériser au mieux le potentiel de ce type de modélisation. Par exemple, plusieurs compétitions de machine learning ont été organisées à cette fin dans le cadre des *Dream Challenges*[3], notamment les éditions *DREAM 4* et *DREAM 8*.

3.1.3 Inférence à partir de données expérimentales

Comme nous l'avons mentionné dans le chapitre 1, plusieurs technologies permettent de collecter des données à partir de mesures effectuées sur un système biologique. De multiples sources de données expérimentales sont utilisables pour l'inférence de réseaux de régulation (Noor et al., 2013). Les données d'expression génétiques, qui témoignent de l'activité des molécules d'ARN messager à l'intérieur des cellules, sont les plus utilisées pour l'inférence des réseaux de régulation puisqu'il s'agit d'un indicateur important sur l'activité d'un système (Delgado et al., 2019). Elles ont ainsi été largement considérées pour la construction de graphes d'influence (Hecker et al., 2009).

3. https://dreamchallenges.org/

Difficultés avec les données expérimentales

Bien que les données expérimentales constituent un élément intéressant pour l'inférence des modèles qualitatifs, plusieurs difficultés se présentent pour leur intégration. Les modèles qualitatifs adoptent généralement une représentation abstraite du système modélisé. Les données génomiques n'étant généralement pas discrètes, il est nécessaire de les discrétiser pour les intégrer à un modèle qualitatif. Une autre difficulté provient de l'aspect bruité des données, en particulier avec les données de séquençage. En effet, la modélisation symbolique constitue un cadre de modélisation idéal où les propriétés sont décrites de manière logique et sans erreurs. Or, les données expérimentales peuvent être trompeuses et il est nécessaire d'adopter une vision *statistique* afin que l'inférence d'un modèle soit robuste au bruit. Par exemple, il est possible qu'un gène exprimé dans une cellule ne soit pas détecté dans le séquençage, et la détection des nucléotides est elle-même également associée à un certain taux d'erreur. L'utilisation de séries temporelles présente une difficulté supplémentaire : il peut être difficile d'abstraire correctement l'intervalle de temps considéré pour les mesures expérimentales. Comme mentionné dans le chapitre 1, une transition d'un modèle qualitatif représente généralement une évolution du système, et n'est pas forcément associé à une durée concrète. Enfin, lorsque des données expérimentales sont combinées à des propriétés dynamiques abstraites pour l'inférence d'un modèle, il est également important de s'assurer que ces deux sources d'informations sont prises en compte de manière équilibrée dans l'inférence [4].

Inférence de modèles qualitatifs à partir de séries temporelles

Les données expérimentales, en particulier les données de séries temporelles, ont été exploitées dans différents travaux pour l'inférence automatique de modèles qualitatifs. Par exemple, (Akutsu et al., 2000) propose une méthode pour l'inférence de réseaux booléens avec prise en compte du bruit des données, à partir de l'énumération des réseaux consistants avec les données. De manière similaire, une méthode est proposée dans (Martin et al., 2007) pour l'inférence de réseaux booléens à partir de données de puces à ADN comportant onze points temporels. La discrétisation employée dans cette dernière méthode repose sur un algorithme de clustering, et l'inférence est réalisée par énumération des modèles consistants avec les données, en prenant notamment en compte les attracteurs. L'approche proposée dans (Ben Abdallah et al., 2017a) pour inférer des réseaux

4. On peut par exemple faire varier l'équilibre entre données expérimentales et propriétés dynamiques et comparer les modèles obtenus.

d'automates avec paramètres temporels à partir de séries temporelles consiste également en l'énumération de l'ensemble des modèles compatibles avec les données. L'inférence est cependant réalisée grâce au langage *ASP*, et la discrétisation est basée sur des seuils. La formulation de l'apprentissage à partir d'un problème d'optimisation a par ailleurs été proposée dans (Barman et al., 2018) à partir de séries temporelles, où l'optimisation est réalisée grâce à un algorithme génétique. Cette stratégie a également été employée dans (Lim et al., 2016) à partir de séries pseudo-temporelles (données *single-cell*), avec l'utilisation de techniques de recherche locale pour l'optimisation du modèle. Ces deux dernières approches ne prennent pas en compte des propriétés dynamiques complexes telles que les attracteurs. Enfin, une approche basée également sur la notion de modèle optimal a été proposée dans le cadre de la programmation logique inductive. Le framework *LFIT* consiste ainsi en l'inférence de programmes logiques représentant la dynamique de systèmes biologiques (Inoue et al., 2014), en utilisant exclusivement des données sous forme de séries temporelles (transitions).

Ces dernières approches basées sur des problèmes d'optimisation et reposant parfois exclusivement sur les données expérimentales peuvent être considérées dans une démarche d'apprentissage automatique, voire d'intelligence artificielle. En effet, elles constituent un changement de paradigme important par rapport à la modélisation manuelle et par rapport aux méthodes basées sur des propriétés dynamiques explicites, en mettant de côté les connaissances a priori sur le système étudié. Ces connaissances peuvent alors être utilisées différemment, par exemple dans un but de validation du modèle.La modélisation y est également simplifiée car il n'est pas nécessaire d'intégrer plusieurs sources de données hétérogènes (par exemple des données expérimentales combinées à des propriétés dynamiques). Établir le lien entre données expérimentales et comportements dynamiques grâce aux méthodes d'apprentissage automatique reste toutefois une tâche ardue. Le framework *LFIT* mentionné précédemment s'inscrit particulièrement dans cette démarche de modélisation automatique. Il se distingue des autres approches en adoptant une méthode d'évaluation locale du modèle sur les données, basée sur la notion de transitions observées du système. C'est donc sur cette approche automatique que nous nous concentrons par la suite. Dans la partie suivante, nous introduisons plus en détail ce framework ainsi que les algorithmes développés dans ce cadre.

3.2 Apprentissage logique pour la modélisation qualitative

Les méthodes d'apprentissage automatique adaptées à la logique, notamment le formalisme de la programmation logique inductive, offrent un cadre très intéressant pour la modélisation qualitative automatique des systèmes biologiques. En particulier, le framework *Learning From Interpretation Transition* (*LFIT*) qui a été spécialement développé pour traiter l'aspect dynamique semble particulièrement adapté en ce sens. C'est donc sur ce formalisme que nous nous concentrons, dans le but de développer une approche spécifiquement adaptée à l'apprentissage à partir de données expérimentales. Nous présentons les concepts généraux de ce framework dans la sous-partie suivante. Nous nous concentrons ensuite sur les hypothèses utilisées au cœur des algorithmes d'apprentissage développés pour *LFIT*, et nous en détaillons leur fonctionnement.

3.2.1 Le framework *LFIT*

La Programmation Logique Inductive est une discipline visant à proposer des méthodes pour l'induction de programmes logiques (Muggleton et al., 1994) à partir d'observations (les données) et de connaissances a priori (*background knowledge*). Cette discipline peut être considérée comme l'association de la programmation logique avec l'apprentissage automatique. Le formalisme logique a l'avantage d'offrir un cadre de modélisation abstrait qui facilite notamment le raisonnement, tandis que l'apprentissage automatique vise à découvrir automatiquement des propriétés à partir de données afin de construire un modèle prédictif. La Programmation Logique Inductive tente ainsi de tirer parti de ces deux disciplines en proposant un cadre formel pour l'induction de modèles.

Le framework *Learning From Interpretation Transition* (*LFIT*), illustré sur la figure 3.1, a été proposé dans ce contexte afin de permettre l'apprentissage d'un modèle dynamique formel (qualitatif) à partir de données présentant un aspect temporel (par exemple des séries temporelles) (Inoue et al., 2014). Ce framework semble donc particulièrement adapté pour l'apprentissage de modèles qualitatifs puisque ceux-ci visent à modéliser l'aspect dynamique du système. Le modèle inféré par *LFIT* est un programme logique multivalué, ce qui offre un cadre de modélisation très proche d'autres formalismes qualitatifs tels que les réseaux booléens ou encore les réseaux d'automates. Un

programme logique inféré par *LFIT* peut ainsi être vérifié avec des méthodes formelles telles que celles abordées dans le chapitre 2. Nous formalisons par la suite les différentes notions concernant la programmation logique multivaluée et l'apprentissage automatique avec *LFIT*. Nous nous basons principalement sur la formalisation effectuée dans (Ribeiro et al., 2018b) car celle-ci est récente et générique. Nous évoquons également par la suite différentes extensions proposées dans le cadre de *LFIT*.

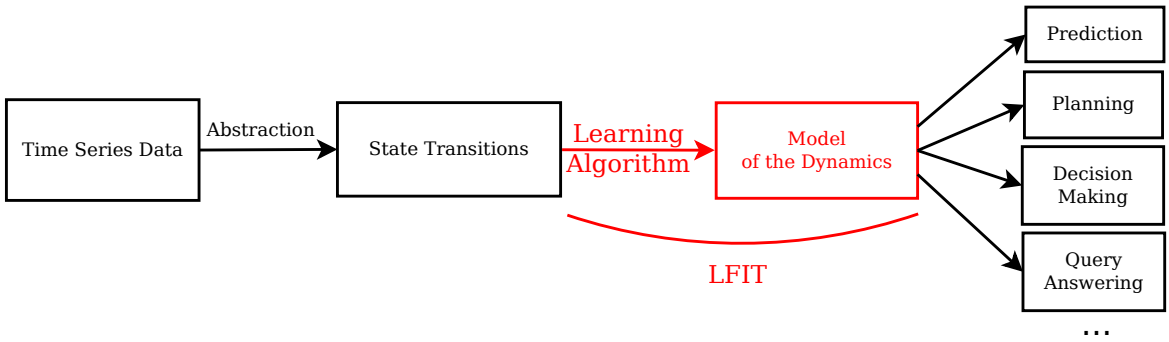

Figure 3.1 – Schéma d'apprentissage logique avec *LFIT*, adapté de (Ribeiro et al., 2018b).

Programme logique multivalué

La logique multivaluée utilisée dans *LFIT* est basée sur la logique annotée, ajoutant ainsi un faible niveau de complexité par rapport à la logique non multivaluée. Nous considérons donc un ensemble de variables discrètes $\mathcal{V}$ d'un système étudié, à partir desquelles des observations dynamiques sont disponibles. Les valeurs discrètes des variables sont représentées de manière logique par des atomes logiques. Les états discrets du système sont ainsi formés à partir d'un atome pour chaque variable, et une transition du système, c'est-à-dire l'évolution du système d'un état vers un nouvel état, est représentée par un couple d'états discrets.

Définition 3.1. *Atome logique, état discret et transition*
Étant donné une variable $v \in \mathcal{V}$ d'un système observable :
- $\mathcal{A}_v = \{v_0, v_1, .., v_k\}$ est l'ensemble des atomes logiques de v, où $\{0, 1, .., k\}$ sont les valeurs discrètes de v.
- $\mathcal{A} = \bigcup_{v \in \mathcal{V}} \mathcal{A}_v$ est l'ensemble total des atomes.
- $\mathcal{S} = \prod_{v \in \mathcal{V}} \mathcal{A}_v$ est l'ensemble des états discrets du système, et $s(v) \in \mathcal{A}_v$ désigne l'atome de la variable v dans l'état s.
- Une transition du système est un couple (s, s') d'états, avec $s, s' \in \mathcal{S}$.

À partir de cette définition, nous pouvons introduire la notion de programme logique qui constitue le modèle inféré par les algorithmes *LFIT*. Les règles logiques dans *LFIT* ont une forme particulière, puisqu'elles visent à décrire l'évolution des variables discrètes du système à partir d'un état donné. Les littéraux qui constituent les règles logiques des programmes considérés dans ce formalisme sont exclusivement des atomes logiques. La tête de la règle (c'est-à-dire sa conclusion) est un atome et correspond à la prédiction de la valeur correspondante pour la variable donnée dans le nouvel état du système. Le corps de la règle, qui désigne la condition logique à vérifier pour son application, est constitué d'un ensemble d'atomes logiques. Celui-ci représente les valeurs des variables devant être vérifiées dans l'état courant du modèle afin de pouvoir en effectuer l'évolution (ou encore la prédiction) qui correspond à la tête de la règle. Ces atomes sont appelés *conditions* dans la suite, désignant ainsi les conditions suffisantes pour que la conclusion de la règle s'applique.

Définition 3.2. *Règle logique multivaluée et programme logique*
Étant donné un ensemble $\mathcal{V}$ de variables discrètes d'un système, et $\mathcal{A}$ l'ensemble des atomes logiques associés :
- Une règle logique multivaluée est définie par un couple $r = (tête(r), corps(r))$, noté $tête(r) \leftarrow corps(r)$ avec $tête(r) \in \mathcal{A}$ désignant la tête de la règle, et $corps(r) \subset \mathcal{A}$ désignant le corps de la règle.
- Un programme logique est défini comme un sous-ensemble de règles logiques multivaluées.

L'application d'une règle à partir d'un état consiste ainsi à vérifier que les atomes présents dans le corps de la règle correspondent aux valeurs des variables concernées dans l'état. Si la vérification est positive, alors nous considérons que la règle **couvre** l'état, et la conclusion de la règle peut être utilisée afin de proposer une prédiction sur l'évolution du système à partir de l'état considéré. La définition suivante formalise la notion de **couverture**.

Définition 3.3. *Couverture règle-état*
Soit r une règle logique et $s \in \mathcal{S}$ un état discret. On dit que la règle r couvre l'état s, noté $couverture(r, s) = \top$, si $\forall v_i \in corps(r), v_i = s(v)$.

À partir de cette dernière définition, nous considérons qu'une règle r **prédit** une transition observée $t = (s, s')$ si $couverture(r, s) = \top$ et si $tête(r) = v_i = s'(v)$. Nous définissons également la notion de domination entre règles. Une règle en domine une autre

si le corps de cette première est inclus dans celui de la deuxième. La notion de domination permet ainsi de caractériser la **généralité** d'une règle, dans le sens où une règle plus générale couvre un nombre plus important d'états discrets. La notion de généralité des règles est à la base des algorithmes d'inférence proposés pour *LFIT*. Cette notion sera discutée par la suite dans le cadre de l'extension que nous proposons.

Définition 3.4. *Domination entre règles*
Soit r et r' deux règles logiques multivaluées. On dit que r domine r', noté $r' \leq r$ si $corps(r) \subseteq corps(r')$ (ce qui implique $\{s \in \mathcal{S} \mid couverture(r', s)\} \subseteq \{s \in \mathcal{S} \mid couverture(r, s)\}$).

L'exemple suivant illustre les différentes notions introduites précédemment.

Exemple 3.1. *Soit $\mathcal{V} = \{a, b, c\}$ et $\mathcal{A} = \{a_0, a_1\} \cup \{b_0, b_1, b_2\} \cup \{c_0, c_1\}$ les variables discrètes et atomes logiques correspondant à un système dynamique observable. Étant donné une transition $t = (s, s')$ observée du système, avec $s = (a_0, b_0, c_1)$ et $s' = (a_1, b_0, c_1)$:*
- La règle $r = a_1 \leftarrow b_0 \wedge c_1$ couvre s, c'est-à-dire $couverture(r, s) = \top$
- r prédit t car $couverture(r, s) = \top$ et $tête(r) = a_1 = s'(a)$
- La règle $r' = a_1 \leftarrow b_0$ domine r, c'est-à-dire $r \leq r'$, impliquant $couverture(r', s) = \top$
- Enfin, la règle $r'' = a_0 \leftarrow b_1$ ne couvre pas s, noté $couverture(r'', s) = \bot$

Étant donné un programme logique multivalué, la simulation du système modélisé s'obtient ainsi via l'application des règles logiques à partir d'un état donné. La question de la sémantique, présentée dans le chapitre 1 se pose alors encore une fois, puisqu'il est nécessaire de décider quelles variables doivent être mises à jour. Cette question est abordée à la fin de ce chapitre.

Par ailleurs, on peut remarquer une grande proximité entre le formalisme de la programmation logique multivaluée introduit ici pour la modélisation qualitative, et celui des réseaux d'automates utilisé dans le chapitre 2. En effet, une règle logique peut être apparentée à une généralisation des transitions locales dans un réseau d'automate, et les atomes logiques peuvent être associés aux états locaux des automates. La règle décrit alors l'évolution de la valeur d'une variable donnée (tête de la règle) en fonction des valeurs d'autres variables dans l'état courant du système (corps de la règle). La différence principale réside dans le fait que dans le cas des réseaux d'automates, l'état local de départ de l'automate est renseigné dans la transition, alors que la valeur courante de la variable

n'est pas forcément contrainte dans une règle logique. Elle est par ailleurs facilement modélisable en rajoutant systématiquement un atome dans le corps de la règle.

Apprentissage automatique à partir de transitions (*LFIT*)

La modélisation proposée par *LFIT* suppose les données en entrée de l'algorithme sous la forme d'un ensemble de transitions $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$. Cette notion de transition est analogue aux arcs des graphes de transitions que nous avons définis dans le chapitre 1, et qui caractérisent les modèles qualitatifs. L'objectif est donc d'inférer les règles d'évolution les plus plausibles à partir des transitions disponibles. Les algorithmes proposés dans le cadre de *LFIT* décomposent généralement ce problème par variable, et par atome au sein de chaque variable. Cela présuppose que la prédiction de l'évolution d'une variable peut être faite indépendamment des prédictions sur les autres variables. La synchronisation des différentes variables est alors considérée comme découlant de la sémantique. L'algorithme 1 illustre ainsi le schéma d'inférence d'un programme logique dans *LFIT* à partir de transitions, avec cette hypothèse d'indépendance.

Données : $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$, un ensemble de transitions, $\mathcal{V}$, $\mathcal{A}$
Résultat : $\mathcal{P}$
$\mathcal{P} \leftarrow \{\}$
pour $v \in \mathcal{V}$ **faire**
  pour $v_i \in \mathcal{A}_v$ **faire**
    $\mathcal{R}_{v_i} \leftarrow \mathit{infererRègles}(v_i, \mathcal{T}_{obs})$
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{R}_{v_i}$
  fin
fin
retourner $\mathcal{P}$

Algorithme 1 : Schéma d'apprentissage dans *LFIT*

La méthode d'inférence de règles pour un atome donné est donc l'élément central des algorithmes proposés pour *LFIT*. Dans ce contexte, le formalisme de la Programmation Logique Inductive considère généralement deux types de données en entrée d'un algorithme d'apprentissage : les exemples positifs, représentant les données à partir desquelles l'atome logique inféré doit pouvoir être déduit, et les exemples négatifs, à partir desquelles l'atome inféré ne doit pouvoir être déduit (Muggleton et al., 1994). Les algorithmes d'inférence de règles pour *LFIT* identifient ces exemples positifs et négatifs en séparant

l'ensemble des états observés en deux sous-ensembles disjoints. Si on considère un atome v_i dont on veut inférer les règles de prédiction à partir des transitions $\mathcal{T}_{obs}$, l'idée est donc de séparer l'ensemble des états observés en première position des transitions (c'est-à-dire $\mathcal{S}_{obs} = \{s \in \mathcal{S} \mid \exists (s, s') \in \mathcal{T}_{obs}\}$) en deux sous-ensembles : les exemples positifs $\mathcal{S}^+_{v_i}$ et les exemples négatifs $\mathcal{S}^-_{v_i}$. La séparation des états peut se faire en fonction de l'observation où non de l'atome v_i dans l'état d'arrivée des transitions, afin d'obtenir un ensemble de règles capable de reproduire les transitions, c'est-à-dire capable de prédire la présence de l'atome v_i à partir d'un état donné si cela a été observé dans les transitions. On définit donc généralement $\mathcal{S}_{obs} = \mathcal{S}^+_{v_i} \uplus \mathcal{S}^-_{v_i}$, où $\mathcal{S}^+_{v_i} = \{s \in \mathcal{S}_{obs} \mid \exists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}$ correspond aux exemples positifs, c'est-à-dire les exemples qui doivent être couverts par les règles inférées, et $\mathcal{S}^-_{v_i} = \mathcal{S}_{obs} \setminus \mathcal{S}^+_{v_i}$ correspond aux exemples négatifs, lesquels ne devant être couverts par aucune règle inférée. L'apprentissage des règles pour un atome donné est ainsi ramené à un problème de classification binaire. Plusieurs méthodes ont été proposées pour traiter ce problème, basées sur un ensemble de propriétés définies pour garantir les ensembles de règles les plus intéressants. Nous détaillons dans la sous-partie 3.2.2 le fonctionnement précis de deux algorithmes en particulier, sur lesquels nous nous appuierons par la suite.

Extensions existantes

Plusieurs extensions ont également été proposées à partir du schéma de modélisation présenté précédemment. Les versions antérieures de ce framework ont été proposées dans un premier temps avec une logique qui n'était pas multivaluée dans (Inoue et al., 2014). L'aspect connaissance a priori était cependant considéré de plusieurs manières, notamment en réduisant la longueur du corps de la règle, et également avec l'utilisation d'une notion de voisinage dans des modèles spatiaux comme les automates cellulaires. Par ailleurs, il est également possible d'imaginer que la prédiction de l'évolution d'un système réel est difficile lorsque celle-ci ne se base que sur le dernier état observé, et non sur les précédents. Une variante nommée *LFkT* a donc été proposée afin d'inclure des observations précédentes dans le corps des règles logiques (Ribeiro et al., 2015), ce que l'on peut apparenter à la modélisation de délais, ou encore à la modélisation d'un système à mémoire. Ensuite, l'aspect probabiliste a été traité dans (Martínez et al., 2015) afin de prédire des comportements non déterministes du modèle. L'algorithme *ACEDIA* a également été proposé dans (Ribeiro et al., 2018a) afin d'inférer des règles logiques à partir de données non discrétisées (les atomes représentent alors des intervalles de valeurs déter-

minés automatiquement). Enfin, la question de la sémantique a été traitée en détails dans (Ribeiro et al., 2018b). Bien que ces extensions semblent intéressantes dans un contexte d'apprentissage à partir de données biologiques, les algorithmes associés ont généralement une haute complexité, et les hypothèses de modélisation sous-jacentes ne sont pas toujours adaptées à ces données particulières, comme on le verra par la suite. Dans ce travail, nous nous restreignons donc à l'utilisation de transitions simples pour l'apprentissage, à partir de données discrétisées et sans notion de délai. Nous nous concentrons en particulier sur l'inférence des règles logiques, l'aspect sémantique étant discuté ultérieurement dans ce chapitre. Enfin, bien que la modélisation probabiliste ait déjà été traitée, nous en proposons par la suite une interprétation différente, adaptée aux données d'expression génétique et qui mènera à l'élaboration d'un nouvel algorithme.

3.2.2 Inférence de règles logiques avec *GULA* et *PRIDE*

Dans cette sous-partie, nous présentons en détails les algorithmes d'inférences de règles logiques *GULA* et *PRIDE* introduits dans le cadre de modélisation *LFIT*, pour l'apprentissage de règles logiques prédisant un atome logique donné. La notion centrale pour l'inférence des règles logiques dans *LFIT* est la notion d'optimalité, définie dans (Ribeiro et al., 2018b). Un programme logique est considéré comme **optimal** pour un ensemble de transition $\mathcal{T}_{obs}$ lorsque ses règles sont aussi générales que possible et permettent de reproduire les transitions données (il n'existe donc pas de règles plus générales expliquant les transitions). Il a également été montré qu'un tel programme logique est unique. Les algorithmes *GULA* et *PRIDE* ont été construits à partir de cette notion d'optimalité. *GULA* est l'algorithme d'inférence *multiusage* permettant l'obtention du programme optimal, mais celui-ci a une complexité exponentielle par rapport au nombre de variables. L'algorithme *PRIDE* a donc été proposé comme une alternative heuristique, ne vérifiant pas formellement la notion d'optimalité. Une implémentation des algorithmes *GULA* et *PRIDE* en langage python (comprenant également les autres algorithmes proposés pour *LFIT*) est disponible en ligne à l'adresse https://github.com/Tony-sama/pylfit).

L'algorithme *GULA* a été introduit dans (Ribeiro et al., 2018b). Il se concentre sur la généralité des règles logiques à inférer. Ainsi, à partir de la séparation des données en exemples positifs $\mathcal{S}^+_{v_i}$ et négatifs $\mathcal{S}^-_{v_i}$ pour un atome v_i donné, l'objectif est d'inférer les règles couvrant un maximum d'états de $\mathcal{S}^+_{v_i}$, tout en ne couvrant aucun des états de $\mathcal{S}^-_{v_i}$. L'ensemble des règles à inférer est d'abord initialisé avec une première règle générique,

couvrant n'importe quel état. Ensuite, les exemples négatifs sont parcourus et dès qu'une des règles inférées couvre un des exemples négatifs, celle-ci est spécialisée en plusieurs règles, de manière à ne plus couvrir l'exemple négatif, tout en restant aussi générale que possible. Cette méthode permet de garantir l'obtention de toutes les explications possibles pour un exemple positif donné. La procédure d'inférence de règles avec *GULA* est formalisée dans l'algorithme 2.

Données : $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$, $v_i \in \mathcal{A}_v$ l'atome à apprendre
Résultat : $\mathcal{R}_{v_i}$ l'ensemble des règles à inférer pour v_i
1 $\mathcal{S}^-_{v_i} \leftarrow \{s \mid \nexists (s, s') \in \mathcal{T}_{obs}, s'(v) = v_i\}$ `// Extraction des exemple négatifs`
2 $\mathcal{R}_{v_i} \leftarrow \{v_i \leftarrow \varnothing\}$ `// Initialisation avec la règle la plus générale`
3 **pour** $s \in \mathcal{S}^-_{v_i}$ **faire**
4 $\quad \mathcal{M}_s \leftarrow \{r \in \mathcal{R}_{v_i} \mid couverture(r, s) = \top\}$
5 $\quad \mathcal{R}_{v_i} \leftarrow \mathcal{R}_{v_i} \setminus \mathcal{M}_s$
6 $\quad$ **pour** $r \in \mathcal{M}_s$ **faire**
7 $\quad\quad \mathcal{R}' \leftarrow specialisation(r, s)$ `// Spécialisation de la règle`
8 $\quad\quad \mathcal{R}' \leftarrow \{r' \in \mathcal{R}' \mid \nexists r \in \mathcal{R}_{v_i} : r' \leq r\}$ `// Filtrage des règles dominées`
9 $\quad\quad \mathcal{R}_{v_i} \leftarrow \{r \in \mathcal{R}_{v_i} \mid \nexists r' \in \mathcal{R}' : r \leq r'\}$ `// Filtrage`
10 $\quad\quad \mathcal{R}_{v_i} \leftarrow \mathcal{R}_{v_i} \cup \mathcal{R}'$
11 $\quad$ **fin**
12 **fin**
13 **retourner** $\mathcal{R}_{v_i}$

Algorithme 2 : Algorithme *GULA*

La spécialisation d'une règle couvrant un exemple négatif est effectuée à la ligne 7, via la fonction *specialisation*. Elle peut être définie de la façon suivante : $specialisation(r, s) = \{tête(r) \leftarrow corps(r) \cup \{v_i\} \mid s(v) \neq v_i \wedge corps(r) \cap \mathcal{A}_v = \varnothing\}$. La spécialisation d'une règle est ainsi obtenue en ajoutant les conditions minimales dans le corps de la règle, afin d'obtenir de nouvelles règles ne couvrant pas l'état, tout en restant aussi générales que possibles (c'est-à-dire que les nouvelles règles couvrent un maximum d'états). Par ailleurs, lors de l'ajout de ces nouvelles règles spécialisées à l'ensemble de règles inférées, un processus de filtrage est nécessaire pour s'assurer qu'il n'y a pas de domination entre ces dernières et les règles déjà calculées. On remarque également que les exemples positifs $\mathcal{S}^+_{v_i}$ ne sont à aucun moment pris en compte dans l'apprentissage. La complexité temporelle de *GULA* est en $\mathcal{O}(|\mathcal{T}_{obs}|^2 + 2n^3d^{2n+1} + 2n^2d^n)$ et sa complexité spatiale est en $\mathcal{O}(d^{2n} + 2d^n + nd^{n+2})$, avec $n = |\mathcal{V}|$ le nombre de variables du système et d le nombre de

valeurs discrètes maximal pour les variables (Ribeiro et al., 2018b). Bien que son temps d'exécution pratique dépende du nombre de règles inféré, son utilisation reste limitée à des exemples de très petite taille.

Données : $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$, $v_i \in \mathcal{A}_v$ l'atome à apprendre
Résultat : $\mathcal{R}_{v_i}$ l'ensemble des règles à inférer pour v_i

$\mathcal{S}^+_{v_i} \leftarrow \{s \mid \exists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}$ `// Extraction des exemple positifs`
$\mathcal{S}^-_{v_i} \leftarrow \{s \mid \exists (s, s') \in \mathcal{T}_{obs}, s \notin \mathcal{S}^+_{v_i}\}$ `// Extraction des exemple négatifs`
$\mathcal{R}_{v_i} \leftarrow \{\}$
tant que $\mathcal{S}^+_{v_i} \neq \varnothing$ **faire**
  $r \leftarrow (v_i \leftarrow \varnothing)$
  choisir $s^+ \in \mathcal{S}^+_{v_i}$
  pour $s^- \in \mathcal{S}^-_{v_i}$ **faire** `// Spécialisation sur les exemples négatifs`
    si $couverture(r, s^-) = \top$ **alors**
      choisir $w_j : s^+(w) = w_j \wedge s^-(w) \neq w_j$
      $r \leftarrow (tête(r) \leftarrow corps(r) \cup \{w_j\})$
    fin
  fin
  pour $w_j \in corps(r)$ **faire** `// Simplification de` r
    $r' \leftarrow (tête(r) \leftarrow corps(r) \setminus \{w_j\})$
    si $\{s^- \in \mathcal{S}^-_{v_j} \mid couverture(r', s^-) = \top\} = \varnothing$ **alors**
      $r \leftarrow r'$
    fin
  fin
  $\mathcal{S}^+_{v_i} \leftarrow \mathcal{S}^+_{v_i} \setminus \{s \mid s \in \mathcal{S}^+_{v_i}, couverture(r, s) = \top\}$ `// Retrait si couverts`
  $\mathcal{R}_{v_i} \leftarrow \mathcal{R}_{v_i} \cup \{r\}$
fin
retourner $\mathcal{R}_{v_i}$

Algorithme 3 : Algorithme *PRIDE*

L'algorithme *PRIDE* a été proposé dans (Ribeiro et al., 2021 ; Ribeiro et al., to appear) principalement pour pallier la complexité importante de *GULA*. Il est basé sur la même notion d'optimalité mais celle-ci n'est pas formellement garantie pour le programme logique qu'il infère. Plus spécifiquement, il a été montré que *PRIDE* infère un sous-ensemble du programme optimal, ce qui signifie que des règles alternatives permettant de reproduire les transitions sont manquantes. Au lieu d'inférer les règles uniquement par spécialisation à partir des exemples négatifs, *PRIDE* itère sur les exemples positifs et infère une règle pour chacun d'entre eux, en spécialisant suffisamment pour que la règle

ne couvre pas d'exemples négatifs. Une règle inférée peut par ailleurs couvrir plusieurs exemples positifs, auquel cas ces derniers sont retirés de l'ensemble des exemples restants à couvrir. L'algorithme 3 synthétise les différentes étapes de l'apprentissage avec *PRIDE*.

La boucle principale de l'algorithme, commençant à la ligne 4, consiste à itérer sur les exemples positifs afin que chacun d'entre eux soit couvert par au moins une règle. La nouvelle règle à inférer est, comme précédemment, initialisée avec un corps vide (règle la plus générale). Des conditions sont ensuite ajoutées afin que la règle ne couvre aucun exemple négatif, tout en s'assurant qu'elle couvre encore l'exemple positif considéré (lignes 9 et 10). Une procédure de simplification de la règle est alors utilisée, dans le cas où des conditions inutiles auraient été ajoutées au corps de celle-ci. Enfin, si d'autres exemples positifs qui n'avaient pas encore été couverts le sont par cette nouvelle règle, ils sont retirés de l'ensemble des exemples positifs à couvrir, et la règle est finalement ajoutée à la liste des règles inférées pour l'atome considéré.

3.3 Extension de *LFIT* pour les données expérimentales

Les algorithmes développés pour le framework *LFIT* le rendent directement applicable sur un ensemble de transitions pour l'apprentissage d'un programme logique. Cependant, la notion d'optimalité sur laquelle ces algorithmes sont basés repose sur des hypothèses qui se révèlent pertinentes dans un cadre formel uniquement. Plus précisément, les notions de bruit et de motifs statistiques dans les données ne sont pas forcément bien traitées avec les algorithmes existants. Dans cette partie, nous discutons en détail des spécificités des données expérimentales rencontrées en biologie, en particulier des données *single-cell* abordées dans le chapitre 4. Nous confrontons ensuite ces données aux hypothèses de modélisation sur lesquelles se base la notion d'optimalité des programmes logiques. À partir de cette confrontation, nous proposons une nouvelle méthode d'inférence, basée sur la formulation d'un problème d'optimisation combinatoire, mieux adapté aux données expérimentales. Cette méthode a été publiée dans (Buchet et al., 2021).

3.3.1 Limites des algorithmes existants

En adoptant le schéma de modélisation proposé par *LFIT* (algorithme 1, page 87), on peut traiter le problème d'inférence comme la construction d'un modèle de classification binaire. En effet, on suppose que pour l'apprentissage d'un atome logique donné, les données peuvent systématiquement être séparées en un ensemble d'exemples positifs $\mathcal{S}^+$, correspondant aux états à partir desquels l'atome a été observé dans les transitions, et un ensemble d'exemples négatifs $\mathcal{S}^-$ à partir desquels l'atome n'a pas été observé. Nous choisissons donc ici de nous abstraire totalement de l'apprentissage des transitions et nous nous concentrons sur l'induction de règles logiques permettant de résoudre au mieux ce problème de classification.

Spécificité des données expérimentales

Dans le contexte de l'apprentissage automatique à partir de données expérimentales, nous proposons plusieurs hypothèses sur la forme des données qui n'ont pas été considérées jusqu'ici dans l'utilisation du framework *LFIT* :

1) On suppose que les données expérimentales sont des données de grande dimension, c'est-à-dire que le nombre de variables $\mathcal{V}$ correspondant est très grand. Cela se justifie par exemple avec l'utilisation des technologies de séquençage mentionnées dans la partie 1.1.2 du chapitre 1 (en page 22), qui permettent d'obtenir l'expression de dizaines de milliers de gènes à partir d'une unique acquisition.

2) L'hypothèse précédente implique que très peu de données sont observées. En effet, une grande dimensionnalité implique que de très nombreux états discrets sont observables en théorie. Il est donc important de prendre en considération les données non observées dans l'inférence de modèle.

3) Nous faisons également l'hypothèse que les données sont particulièrement bruitées, c'est-à-dire que les états discrets observés comportent des erreurs. En particulier, il est intéressant de prévoir que la présence d'un 0 dans les données peut signifier qu'une variable (par exemple un gène) n'a pas été détectée alors qu'elle était réellement présente. Cet aspect bruité se justifie notamment par l'utilisation de technologies de séquençage *single-cell* considérées dans le chapitre 4.

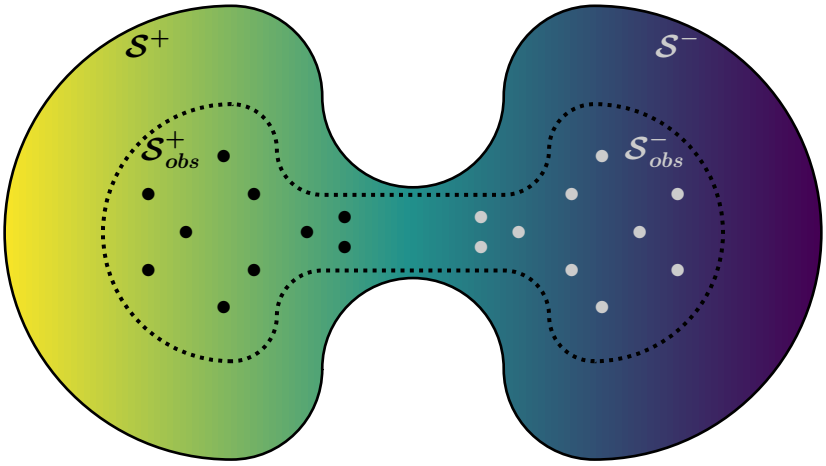

FIGURE 3.2 – Représentation intuitive des exemples positifs et négatifs pour l'inférence d'un atome logique. Les exemples positifs observés sont représentés dans la zone pointillée à gauche et les exemples négatifs sont représentés dans la zone pointillée à droite.

4) Enfin, il peut être intéressant d'interpréter les données comme un continuum, c'est-à-dire que pour une donnée particulière, il est toujours possible de trouver une donnée similaire, identique sur un nombre important de variables. Cette hypothèse peut être étendue en supposant qu'il existe certains exemples positifs de $\mathcal{S}^+$ similaires à certains exemples négatifs de $\mathcal{S}^-$. Encore une fois, cette hypothèse est spécifique aux données de séquençage *single-cell* et se justifie par le fait qu'un échantillon contient des cellules distribuées de manière homogènes par rapport à l'ensemble des cellules observables, et qu'au sein d'un type cellulaire donné, les cellules sont très similaires et on n'observe pas de sous-échantillon isolé.

La figure 3.2 propose une représentation visuelle intuitive des exemples positifs et négatifs pour une instance d'inférence de règle donnée. Cette figure représente l'ensemble des données observables pour un problème d'inférence donné, avec une séparation entre les exemples positifs $\mathcal{S}^+$ (à gauche, en jaune) et les exemples négatifs $\mathcal{S}^-$ (à droite, en bleu foncé). La zone pointillée délimite l'ensemble des données qui sont effectivement observées, également décomposées en deux sous ensembles $\mathcal{S}^+_{obs}$ et $\mathcal{S}^-_{obs}$. Ainsi, chaque point représente un état discret donné pour l'apprentissage, appartenant aux exemples positifs (en noir) ou bien aux exemples négatifs (en gris). Nous représentons donc le fait que les données observées constituent seulement un sous-ensemble des données observables : $(\mathcal{S}^+_{obs} \cup \mathcal{S}^-_{obs}) \subsetneq (\mathcal{S}^+ \cup \mathcal{S}^-)$. On suppose également qu'il existe des exemples positifs et des

exemples négatifs qui se ressemblent, ce qui est représenté par la connexion entre les deux ensembles d'exemples positifs et négatifs, et par le dégradé de couleur.

Défauts des algorithmes existants

Étant donné les différentes hypothèses énoncées précédemment, nous identifions deux difficultés principales avec les algorithmes d'inférence de règles existants pour le traitement des données expérimentales :

La **première** difficulté est liée à la définition de programme logique optimal établie dans (Ribeiro et al., 2018b), et utilisée pour guider l'inférence dans les algorithmes *GULA* et *PRIDE*. En effet, cette définition pousse à inférer les règles les plus générales possibles, c'est-à-dire couvrant un maximum d'états non négatifs. Cependant, nous supposons ici que beaucoup de données ne sont pas observées, donc que beaucoup d'exemples négatifs ne sont pas connus [5]. Il semble ainsi risqué d'inférer les règles les plus générales car l'ensemble de règles résultant risque de mener à une prédiction erronée d'un atome dans beaucoup de cas.

La **deuxième** difficulté provient directement du formalisme de programmation logique utilisé pour le framework *LFIT*. En effet, ce formalisme n'est pas tolérant au bruit des données et la définition de couverture règle-état peut compliquer l'inférence. Une règle ne peut ainsi pas couvrir un état donné si cet état comporte au moins une erreur par rapport au corps de la règle. La présence de bruit complique donc la recherche de règles générales, c'est-à-dire de règles couvrant un maximum d'états.

Nous illustrons ces deux difficultés sur un système de très petite dimension avec l'exemple qui suit.

Exemple 3.2. *Soit $\mathcal{V} = \{a, b\}$ un ensemble de variables, et $\mathcal{A} = \{a_0, a_1\} \cup \{b_0, b_1\}$ un ensemble d'atomes associés pour un système donné :*
- Supposons que pour l'inférence d'un atome donné, on ait $\mathcal{S}^+_{obs} = \{(a_0, b_1)\}$ et $\mathcal{S}^-_{obs} = \{(a_1, b_0)\}$ des données bruitées observées expérimentalement. Alors, la règle $r = \text{tête}(r) \leftarrow b_1$ est cohérente avec $\mathcal{S}^+_{obs}$ et $\mathcal{S}^-_{obs}$ car $couverture(r, (a_0, b_1)) = \top$ et $couverture(r, (a_1, b_0)) = \bot$. Cette règle ne peut pas être généralisée car le retrait de la condition la ferait couvrir

5. On peut raisonner de manière générale par rapport à cela. Un algorithme d'apprentissage sera toujours biaisé par les données fournies en entrée et il faudrait une diversité importante de données.

l'exemple négatif, r fait donc partie du programme optimal pour ce problème et elle serait retournée par GULA. Par ailleurs, l'état (a_1, b_1) *n'a pas été observé, et on a donc* $couverture(r, (a_1, b_1)) = \top$*, ce qui peut être problématique si* (a_1, b_1) *est en réalité un exemple négatif.*
- Maintenant, plaçons-nous dans le cas où deux ensembles d'exemples positifs et négatifs alternatifs $\mathcal{S}^{+'}_{obs}$ *et* $\mathcal{S}^{-'}_{obs}$ *sont donnés, avec* $\mathcal{S}^{+'}_{obs} = \{(a_0, b_0), (a_1, b_1)\}$ *et* $\mathcal{S}^{-'}_{obs} = \{(a_1, b_0), (a_0, b_1)\}$*. Supposons également que a soit bruité, et donc les données non bruitées seraient telles que :* $\mathcal{S}^{+'}_{réel} = \{(a_0, b_0), (a_0, b_1)\}$ *et* $\mathcal{S}^{-'}_{réel} = \{(a_1, b_0), (a_1, b_1)\}$*. Avec la règle* $r' = tête(r') \leftarrow a_0$*, on a alors* $couverture(r', s^+) = \top \quad \forall s^+ \in \mathcal{S}^{+'}_{réel}$ *et* $couverture(r', s^-) = \bot \quad \forall s^- \in \mathcal{S}^{-'}_{réel}$*. La règle* r' *peut alors généraliser sur les exemples positifs réels sans risquer de couvrir des exemples négatifs car* a_0 *semble caractériser les exemples positifs. Cependant, à cause de la présence du bruit,* r' *ne généralise pas sur les données observées et il est donc difficile d'inférer une telle règle en se basant uniquement sur la définition de la couverture.*

On peut alors se demander quelle stratégie serait la plus adéquate pour l'inférence de règles sous les hypothèses présentées en début de cette partie. En reprenant la représentation schématique des données proposée dans la figure 3.2, on peut tenter de représenter le comportement de différentes stratégies d'inférence de règles. La figure 3.3 illustre ainsi les ensembles de règles obtenus par trois stratégies d'inférences différentes. Sur cette figure, un cercle violet représente une règle logique, et les états couverts par cette règle sont représentés par les points couverts par le cercle. La stratégie **Générale** correspond donc à l'inférence des règles les plus générales, c'est-à-dire couvrant un maximum d'états non négatifs. C'est la stratégie adoptée par *GULA*, et on peut remarquer que cette stratégie pousse les règles à couvrir beaucoup d'exemples négatifs non connus. La deuxième stratégie, que l'on appelle stratégie **Spécifique**, consiste à inférer les règles les plus spécifiques possibles. Ainsi, une unique règle est inférée pour chaque exemple positif, dont le corps contient tous les atomes présents dans l'exemple. Cette stratégie ne permet pas d'obtenir un modèle généralisant sur les exemples positifs, ce qui peut être interprété comme une situation d'*overfitting*. Enfin, une dernière stratégie nommé stratégie **Généralisante** consiste à inférer des règles généralisant sur les exemples positifs, mais pas au-delà. Avec cette stratégie, chaque règle couvre plusieurs exemples positifs, et les règles ne généralisent pas à ce qui n'est pas observé. C'est la stratégie qui semble la plus adéquate selon les hypothèses que nous avons formulées, et c'est donc celle que nous cherchons à obtenir.

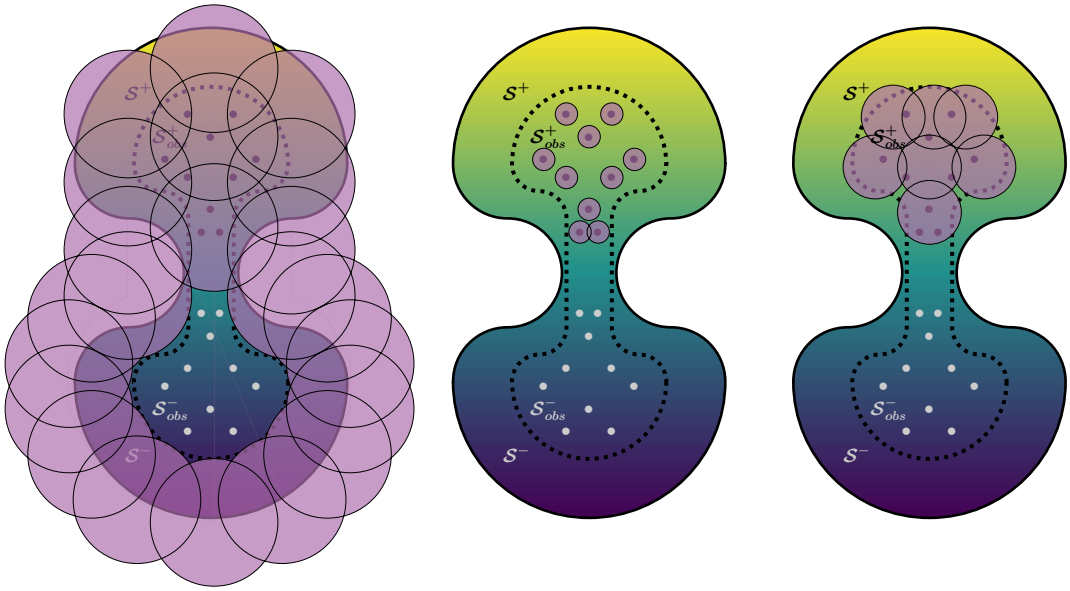

FIGURE 3.3 – Représentation de différentes stratégies d'apprentissage de règles. Chaque cercle représente une règle logique. Le schéma de gauche représente les règles les plus générales, celui du milieu représente les règles les plus spécifiques, et enfin celui de droite représente des règles généralisant sur les exemples positifs.

Vers une nouvelle méthode d'inférence

La **généralisation** des règles à travers les exemples positifs nous semble donc être un aspect particulièrement important à considérer pour l'inférence d'un modèle. En effet, puisqu'il est risqué de considérer les exemples non observés comme des exemples positifs, on peut supposer que chercher les points communs entre les exemples positifs, et leurs différences avec les exemples négatifs, est une alternative pour généraliser les règles logiques. Le contexte de modélisation logique complique par ailleurs cette généralisation puisque la présence du bruit empêche une même règle de couvrir complètement plusieurs états. Pour y remédier, il est donc intéressant de relaxer la définition de couverture entre une règle et un état discret. Un compromis semble par ailleurs nécessaire par rapport au nombre de règles à inférer dans ce contexte. En effet, il est peu judicieux de proposer une règle par exemple positif, puisque cela ne permet pas de généraliser. Le cas opposé, qui consiste à inférer une unique règle pour tous les exemples positifs, est également difficile à mettre en œuvre car une unique règle ne peut vraisemblablement pas caractériser efficacement tous les exemples positifs d'un jeu de données.

3.3.2 Extension par optimisation combinatoire

Afin de proposer une méthode d'inférence adéquate pour les données considérées dans cette thèse, on se concentre donc sur la généralisation des règles logiques entre les différents exemples positifs. Comme expliqué précédemment, le bruit des données empêche une règle de couvrir plusieurs exemples positifs à la fois, car il est peu probable que les atomes du corps de la règle soient tous présents simultanément dans plusieurs états. Par ailleurs, on peut espérer que des motifs récurrents soient détectables dans les exemples positifs. Nous proposons donc de raffiner la définition de couverture entre une règle et un état afin de faire la différence entre le cas où aucun atome de la règle n'est présent dans l'état, et le cas où seulement un atome n'est pas présent. Nous introduisons ainsi la définition suivante de l'erreur de couverture entre une règle et un état, qui permet de quantifier la couverture entre une règle et un état.

Définition 3.5. *Erreur de couverture règle-état*
Soit un ensemble $\mathcal{V}$ de variables et un ensemble $\mathcal{A}$ d'atomes pour un ensemble $\mathcal{S}$ d'états. Soit r une règle logique, et $s \in \mathcal{S}$ un état discret.
L'erreur de couverture de r sur s, noté $erreurCouverture(r, s)$ est définie par :
$erreurCouverture(r, s) = |\{v_i \in corps(r) \mid v_i \neq s(v)\}| = \sum_{v_i \in corps(r)} \left(1 - \mathbb{1}_{\{s(v)\}}(v_i)\right)$
où $\mathbb{1}_{\mathcal{A}}(x) = 1$ si $x \in \mathcal{A}, 0$ sinon.

Il est par ailleurs également possible d'établir un lien entre l'erreur de couverture définie ici, et la couverture d'un état par une règle introduite dans la définition 3.3. En effet, une erreur de couverture d'une règle r inférieure à un entier k sur un état donné signifie qu'il existe une sous règle r' couvrant cet état et telle que $corps(r') \subset corps(r)$.

Formulation d'un problème d'optimisation

La définition 3.5 permet donc de raffiner l'évaluation d'une règle sur des états bruités. Intuitivement, puisque les règles doivent couvrir au mieux les exemples **positifs**, il nous paraît donc judicieux d'inférer la règle logique qui **minimise** globalement l'erreur de couverture sur cet ensemble d'états. De manière similaire, il semble intéressant de prendre le moins de risques possibles de couvrir des exemples **négatifs**, en particulier en présence de bruit. Cela nous pousse donc à chercher à **maximiser** l'erreur sur ces exemples. Ces deux critères nous conduisent ainsi à reformuler le problème de l'inférence de règles sous forme d'un problème d'optimisation à deux objectifs. En se concentrant sur l'inférence

d'une unique règle optimale, et en représentant la généralisation entre les exemples avec l'utilisation d'un score moyen, on peut ainsi formuler le problème d'optimisation biobjectif suivant (les symboles en gras représentent les variables à optimiser, c'est-à-dire le corps des règles logiques) :

$$\begin{aligned} &\text{- } \min_{\boldsymbol{r}} \quad \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} erreurCouverture(\boldsymbol{r}, s^+) \\ &\text{- } \max_{\boldsymbol{r}} \quad \frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} erreurCouverture(\boldsymbol{r}, s^-) \end{aligned} \tag{3.1}$$

Cependant, la résolution d'un problème d'optimisation à deux objectifs ne donne pas lieu à un unique optimum mais à plusieurs solutions *pareto-optimales*, c'est-à-dire incomparables entre elles, car donnant des poids différents aux deux critères optimisés. Dans le cas de l'inférence de règles, il est difficile de déterminer s'il est plus important d'optimiser sur les exemples positifs ou sur les exemples négatifs. En l'absence d'hypothèses supplémentaires, nous proposons alors de simplifier le problème en le réduisant à un seul objectif, à savoir en maximisant la différence entre la moyenne de l'erreur de couverture sur les exemples négatifs et celle sur les exemples positifs. La définition suivante introduit ainsi un score global sur les données, applicable à toute règle logique.

Définition 3.6. *Score d'une règle logique*
Soit $\mathcal{V}$ un ensemble de variables, $\mathcal{A}$ l'ensemble des atomes logiques associés, et $\mathcal{S}^+$ et $\mathcal{S}^-$ des ensembles d'exemples positifs et négatifs pour un problème d'inférence de règle logique. Le score global d'une règle $\boldsymbol{r}$ sur $\mathcal{S}^+$ et $\mathcal{S}^-$, noté $score(\boldsymbol{r})$ est défini par :
- $score(\boldsymbol{r}) = \frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} erreurCouverture(\boldsymbol{r}, s^-) - \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} erreurCouverture(\boldsymbol{r}, s^+)$
On peut noter que $score(\boldsymbol{r}) \in [-|corps(\boldsymbol{r})|, |corps(\boldsymbol{r})|]$.

Développement et détermination de la règle optimale

La définition 3.6 permet de se concentrer sur la recherche d'une (seule) règle optimale pour $\mathcal{S}^+$ et $\mathcal{S}^-$ (la question de l'inférence de plusieurs règles est abordée par la suite). On peut dans un premier temps remarquer que l'ensemble des règles possibles est énumérable, et donc que l'optimisation d'un tel score est possible. À partir de la définition 3.6, on peut développer le terme $erreurCouverture(r, s)$ afin d'expliciter le score à optimiser en fonction des atomes présents dans le corps de la règle. Le score d'une règle sur les exemples positifs et négatifs peut ainsi être développé de la manière suivante :

$$
\begin{aligned}
score(\boldsymbol{r}) &= \frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} erreurCouverture(\boldsymbol{r}, s^-) - \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} erreurCouverture(\boldsymbol{r}, s^+) \\
&= \frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} \sum_{v_i \in corps(\boldsymbol{r})} \left(1 - \mathbb{1}_{\{s^-(v)\}}(v_i)\right) - \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} \sum_{v_i \in corps(\boldsymbol{r})} \left(1 - \mathbb{1}_{\{s^+(v)\}}(v_i)\right) \\
&= \frac{1}{|\mathcal{S}^-|} \sum_{v_i \in corps(\boldsymbol{r})} \sum_{s^- \in \mathcal{S}^-} \left(1 - \mathbb{1}_{\{s^-(v)\}}(v_i)\right) - \frac{1}{|\mathcal{S}^+|} \sum_{v_i \in corps(\boldsymbol{r})} \sum_{s^+ \in \mathcal{S}^+} \left(1 - \mathbb{1}_{\{s^+(v)\}}(v_i)\right) \\
score(\boldsymbol{r}) &= \sum_{v_i \in corps(\boldsymbol{r})} \left(\frac{1}{|\mathcal{S}^-|} \sum_{s^- \in \mathcal{S}^-} \left(1 - \mathbb{1}_{\{s^-(v)\}}(v_i)\right) - \frac{1}{|\mathcal{S}^+|} \sum_{s^+ \in \mathcal{S}^+} \left(1 - \mathbb{1}_{\{s^+(v)\}}(v_i)\right) \right)
\end{aligned}
$$

Le score reformulé ainsi met en valeur l'indépendance des atomes du corps de la règle. En effet, on remarque que le score d'une règle r peut être obtenu à partir d'une somme de valeurs calculées indépendamment pour chaque atome. La partie entre parenthèses ne dépend que d'un seul atome logique v_i, et des ensembles d'exemples positifs $\mathcal{S}^+$ et négatifs $\mathcal{S}^-$. Afin de mieux formaliser cette décomposition du score d'une règle logique, nous introduisons dans la définition suivante l'erreur positive, l'erreur négative, ainsi que le score d'un atome. L'erreur positive et l'erreur négative d'un atome v_i correspondent respectivement aux nombres d'états dans $\mathcal{S}^+$ et $\mathcal{S}^-$ dans lesquels l'atome logique n'est pas vérifié pour la variable v ($s(v) \neq v_i$), représentant intuitivement l'erreur de prédiction résultante de l'utilisation de cet atome sur les données. Nous reprenons ainsi la terminologie utilisée pour les règles car le score d'un atome dépend à la fois de $\mathcal{S}^+$ et $\mathcal{S}^-$, tandis que l'erreur positive et l'erreur négative ne dépendent respectivement que de $\mathcal{S}^+$ ou de $\mathcal{S}^-$.

Définition 3.7. *Erreur positive, erreur négative, et score d'un atome logique*
Soit $\mathcal{V}$ un ensemble de variables, $\mathcal{A}$ un ensemble d'atomes logiques associés à $\mathcal{V}$, et $\mathcal{S}^+$ et $\mathcal{S}^-$ des ensembles d'exemples positifs et négatifs associés à un problème d'inférence de règle. Étant donné un atome $v_i \in \mathcal{A}$:
- L'erreur positive $erreur^+ \in [\![0, |\mathcal{S}^+|]\!]$ de v_i sur $\mathcal{S}^+$ et l'erreur négative $erreur^- \in [\![0, |\mathcal{S}^-|]\!]$ de v_i sur $\mathcal{S}^-$ sont définies par :

$$
\begin{aligned}
erreur^+(v_i) &= \sum_{s^+ \in \mathcal{S}^+} \left(1 - \mathbb{1}_{\{s^+(v)\}}(v_i)\right) \\
erreur^-(v_i) &= \sum_{s^- \in \mathcal{S}^-} \left(1 - \mathbb{1}_{\{s^-(v)\}}(v_i)\right)
\end{aligned}
$$

- Le score de l'atome v_i ($\in [-1, 1]$) est donné par :

$$score(v_i) = \frac{erreur^-(v_i)}{|\mathcal{S}^-|} - \frac{erreur^+(v_i)}{|\mathcal{S}^+|}$$

Cette dernière définition permet de réécrire le score d'une règle de la manière suivante :

$$score(\boldsymbol{r}) = \sum_{\boldsymbol{v_i} \in \boldsymbol{corps(r)}} score(\boldsymbol{v_i}) \tag{3.2}$$

Règle logique (bornée) optimale

La reformulation du score donnée par l'expression 3.2 simplifie la recherche de règles optimales. Une règle optimale peut en effet être obtenue en calculant indépendamment le score de chaque atome, et en construisant la règle contenant les atomes ayant le meilleur score. Cependant, la définition 3.6 implique que $score(r) \in [-|corps(r)|, |corps(r)|]$. Les scores ne sont donc pas comparables pour des règles qui ne possèdent pas le même nombre d'atomes dans leur corps. Afin de simplifier le problème dans un premier temps, nous fixons le nombre d'atomes des règles à optimiser à une constante $k \in \mathbb{N}$. Le choix de cette constante sera discuté par la suite. On obtient alors le problème d'optimisation suivant pour les règles logiques :

$$\max_{\boldsymbol{r}, |corps(\boldsymbol{r})|=k} \sum_{\boldsymbol{v_i} \in corps(\boldsymbol{r})} score(\boldsymbol{v_i}) \tag{3.3}$$

Ainsi, lorsqu'on fixe le nombre d'atomes à k, la règle optimale selon la définition 3.6 peut être calculée en triant les atomes selon leurs scores et en sélectionnant les k atomes ayant les plus grands scores [6]. Nous formalisons la propriété de règle optimale étant donné k dans la proposition qui suit.

Proposition 3.1. *Règle logique optimale*
Soit $\mathcal{V}$ un ensemble de variables discrètes, $\mathcal{A}$ les atomes logiques associés, $\mathcal{S}^+$, $\mathcal{S}^-$ des ensembles d'exemples positifs et négatifs pour un problème d'inférence de règles, et $k \in \mathbb{N}$ une constante représentant le nombre d'atomes dans le corps des règles.
Soit r^* la règle obtenue en sélectionnant les k meilleurs atomes selon leur score.
- Alors $\max_{\boldsymbol{r}:|corps(\boldsymbol{r})|=k} score(\boldsymbol{r}) = score(r^*)$, c'est-à-dire r^* est une règle de longueur k optimale pour $\mathcal{S}^+$ et $\mathcal{S}^-$.

6. S'il y a des égalités au rang k, l'ensemble des règles optimales peut être obtenu en sélectionnant des sous-ensembles d'atomes égaux, en fonction du nombre k déterminé.

Démonstration. Supposons que r^* ne soit pas optimale. Alors il existe une règle r', avec $|corps(r')| = k$ et telle que $score(r') > score(r^*)$.
On a donc $\sum_{v_i \in corps(r')} score(v_i) > \sum_{v_i \in corps(r^*)} score(v_i)$
$\implies \exists v_i \in corps(r') : \forall w_j \in corps(r^*) : score(v_i) > score(w_j)$
Or, ceci est impossible car r^* a été formée avec les k atomes ayant les plus grands scores. Il y a donc contradiction. □

Cette dernière proposition permet donc la mise en place d'une procédure simple pour le calcul de la règle logique optimale selon la définition 3.6, étant donné une constante k fixée. Cependant, il est difficile de choisir la valeur de k. D'un point de vue général, si k est grand, la règle inférée comporte alors beaucoup d'atomes. Bien qu'une telle règle risque peu de couvrir les exemples négatifs, elle couvre également difficilement les exemples positifs, ce qui entraîne la dégradation de son score. D'un autre côté, une règle avec peu d'atomes (dans son corps) est générale et obtient donc un bon score sur les exemples positifs mais un mauvais score sur les exemples négatifs.

Algorithme final de *LOLH* et implémentation

Afin de mieux comprendre comment paramétrer le choix du nombre d'atomes k pour l'inférence d'une règle, on peut se demander comment évolue le score de la règle lorsque ce paramètre varie. Le score dépendant du nombre d'atomes (c'est-à-dire $score(r) \in [-|corps(r)|, |corps(r)|]$), il est possible de le normaliser par la longueur de la règle pour comparer des règles de longueur différente : $score'(r) = \frac{score(r)}{|corps(r)|}$. Cependant, avec cette définition alternative, le score normalisé d'une règle ne peut que décroître par l'ajout d'atomes dans le corps de cette règle. En effet, puisque les atomes sont triés selon leur score, les atomes ajoutés dans le corps d'une règle ont toujours un score inférieur ou égal au score normalisé de la règle. Par exemple, considérons deux règles r_1 et r_2, avec $corps(r_1) = \{v_i\}$, et $corps(r_2) = \{v_i, w_j\}$. En supposant que $score(w_j) < score(v_i)$, on a alors $score(w_j) = score(v_i) - \delta$ avec $\delta \geq 0$. On a donc $score'(r_1) = score(v_i)$ et $score'(r_2) = \frac{score(v_i)+score(w_j)}{2} = \frac{2*score(v_i)-\delta}{2} = score(v_i) - \frac{\delta}{2} \leq score'(r_1)$. Bien qu'en apparence la règle contenant un unique atome semble ainsi être la règle optimale, il n'est pas pertinent du point de vue des données d'utiliser cette règle dans un modèle qualitatif. En effet, l'utilisation d'une seule variable pour effectuer la prédiction rend le modèle très sensible au bruit, et donc il est plus susceptible de faire des erreurs de prédiction sur des données non observées. On peut associer ceci au phénomène de l'*overfitting* mentionné

précédemment. L'utilisation de plusieurs atomes, tant que ceux-ci permettent de faire une prédiction cohérente avec les données, apporte une plus grande robustesse au modèle.

En plus du nombre d'atomes, le score de ceux-ci doit aussi être considéré. En effet, la sélection d'un atome avec un mauvais score (par exemple proche de 0) risque de dégrader les prédictions d'une règle, comme nous l'illustrons dans l'exemple jouet présenté dans la partie 3.3.3. Ainsi, au lieu de se concentrer sur le nombre d'atomes à sélectionner, nous proposons l'utilisation d'un seuil t sur les scores des atomes. L'inférence de règles consiste alors à créer la règle dont le corps contient tous les atomes qui ont un score supérieur ou égal à ce seuil : $corps(r^*) = \{v_i \in \mathcal{A} \mid score(v_i) \geq t\}$, ce qui en fait une méthode simple et rapide. L'algorithme 4, que nous nommons *Learning Optimized Logical Hypothesis* (*LOLH*), résume les différentes étapes de calculs permettant l'inférence d'une règle optimale. Nous proposons une double implementation de cet algorithme en *python* et en *C++*, disponible en ligne à l'adresse http://doi.org/10.5281/zenodo.4738850 et également à l'adresse https://github.com/smbct/LOLH.git (version en développement).

Données : $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$, $v_i \in \mathcal{A}_v$ l'atome à apprendre, $t \in [-1, 1]$ un seuil
Résultat : $\mathcal{R}_{v_i}$ l'ensemble des règles à inférer pour v_i

$\mathcal{S}^+_{v_i} \leftarrow \{s \mid \exists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}$ `// Extraction des exemple positifs`

$\mathcal{S}^-_{v_i} \leftarrow \{s \mid \exists (s, s') \in \mathcal{T}_{obs}, s \notin \mathcal{S}^+_{v_i}\}$ `// Extraction des exemple négatifs`

pour $w_j \in \mathcal{A}$ **faire** `// Calcul du score de chaque atome`

  $erreur^+(w_j) \leftarrow \sum_{s^+ \in \mathcal{S}^+_{v_i}} \left(1 - \mathbb{1}_{\{s^+(w)\}}(w_j)\right)$

  $erreur^-(w_j) \leftarrow \sum_{s^- \in \mathcal{S}^-_{v_i}} \left(1 - \mathbb{1}_{\{s^-(w)\}}(w_j)\right)$

  $score(w_j) \leftarrow \frac{erreur_-(w_j)}{|\mathcal{S}^-|} - \frac{erreur_+(w_j)}{|\mathcal{S}^+|}$

fin

$r^* \leftarrow (v_i \leftarrow \{w_j \in \mathcal{A} \mid score(w_j) \geq t\})$ `// Création de la règle optimale`

$\mathcal{R}_{v_i} \leftarrow \{r^*\}$

retourner $\mathcal{R}_{v_i}$

Algorithme 4 : Algorithme *LOLH*

Extension à plusieurs règles logiques

La méthode *LOLH* permet donc d'inférer une unique règle logique tentant d'optimiser sa prédiction sur l'ensemble total des exemples positifs et des exemples négatifs. On peut

alors se demander comment étendre cette méthode à l'inférence de plusieurs règles logiques afin d'améliorer les performances de prédiction du modèle. Intuitivement, augmenter le nombre de règles permet d'augmenter la diversité des prédictions du modèle. On peut alors imaginer que l'exemple extrême est celui décrit dans la stratégie "Spécifique" de la figure 3.3 en page 97, pour lequel une règle est inférée pour chaque exemple positif. Un compromis entre l'inférence d'une unique règle, et l'inférence d'une règle par exemple positif consisterait donc à inférer plusieurs règles telles que chacune des règles permettent de couvrir un sous-ensemble des exemples positifs. Le nombre de règles contrôlerait alors la spécificité du modèle : un nombre important de règles conduirait à l'*overfittig* tandis qu'un nombre trop faible de règles donnerait un modèle avec des prédictions sous-optimales. Plusieurs solutions peuvent être envisagées pour l'inférence d'un tel modèle. Une solution pourrait consister en la recherche d'une partition optimale sur l'ensemble des exemples positifs, de manière à calculer une règle optimale par sous-ensemble de cette partition. Le calcul d'une telle partition pourrait être effectué grâce à la formulation d'un problème d'optimisation. On peut également envisager l'utilisation de méthodes de clustering afin de partitionner automatiquement les exemples positifs selon leurs similarités.

3.3.3 Exemple jouet

Nous illustrons ici l'utilisation de notre algorithme *LOLH* pour inférer une règle logique optimale sur un petit exemple. Cet exemple contient des états discrets observés pour 4 variables possédant 2 valeurs discrètes chacune : $\mathcal{V} = \{a, b, c, d\}$. Il y a donc un total de 8 atomes logiques pour ces données : $\mathcal{A} = \{a_0, a_1\} \cup \{b_0, b_1\} \cup \{c_0, c_1\} \cup \{d_0, d_1\}$. 8 états discrets sont donnés, notés s_0 à s_7, et séparés en exemples positifs et négatifs de la manière suivante : $\mathcal{S}^+ = \{s_0, s_1, s_2, s_3\}$ et $\mathcal{S}^- = \{s_4, s_5, s_6, s_7\}$. Afin de pouvoir inférer une règle logique sur cet exemple avec la méthode *LOLH*, nous fixons le seuil $t = 0.4$ pour la sélection des atomes à partir de leur score sur $\mathcal{S}^+$ et $\mathcal{S}^-$. Les tableaux suivants contiennent les valeurs discrètes des variables dans chaque état discret observé (les lignes correspondant aux états et les colonnes correspondant aux variables).

	a	b	c	d
s_0	1	0	0	0
s_1	1	0	1	0
s_2	1	0	1	1
s_3	0	0	1	0

TABLE 3.1 – Exemples positifs.

	a	b	c	d
s_4	1	1	1	1
s_5	0	1	0	1
s_6	1	1	0	1
s_7	1	0	0	1

TABLE 3.2 – Exemples négatifs.

On peut remarquer que les variables b et d sont majoritairement égales à 0 dans les exemples positifs, et elles sont majoritairement égales à 1 dans les exemples négatifs. On peut aussi remarquer que c se comporte globalement de manière inverse. Ainsi, on peut s'attendre à ce que les atomes logiques b_0, c_1 et d_0 aient les plus hauts scores. Au contraire, la variable a n'a pas de valeur dominante entre les exemples positifs et les exemples négatifs, ainsi a_0 et a_1 ne devraient pas être sélectionnés pour former une règle logique.

Calcul des scores des atomes

Pour chaque atome, on peut calculer facilement l'erreur positive $erreur^+$ sur $\mathcal{S}^+$ et l'erreur négative $erreur^-$ sur $\mathcal{S}^-$, ainsi que le score à partir de la définition 3.7 (page 100). Par exemple, le score de l'atome b_0 est obtenu de la manière suivante :

— $erreur^+(b_0) = |\{s_i, \mathbf{i} \in [\![\mathbf{0}, \mathbf{3}]\!] \mid s_i(b) \neq b_0\}| = 0$
— $erreur^-(b_0) = |\{s_i, \mathbf{i} \in [\![\mathbf{4}, \mathbf{7}]\!] \mid s_i(b) \neq b_0\}| = 3$
— $score(b_0) = \frac{erreur^-(b_0)}{|\mathcal{S}^-|} - \frac{erreur^+(b_0)}{|\mathcal{S}^+|} = 3/4 - 0/4 = 0.75$

	$erreur^+$	$erreur^-$	$score$
a_0	3	3	0
a_1	1	1	0
b_0	0	3	0.75
b_1	4	1	-0.75
c_0	3	1	-0.5
c_1	1	3	0.5
d_0	1	4	0.75
d_1	3	0	-0.75

TABLE 3.3 – Erreur positive, erreur négative et score pour chaque atome.

Le tableau 3.3 donne les erreurs positives, les erreurs négatives, et les scores de tous les atomes pour cet exemple. Les erreurs positives et négatives de chaque atome sont également représentées en figure 3.4 en page 106 sur un graphique à 2 dimensions (avec l'erreur positive des atomes en abscisse et l'erreur négative en ordonnée). Puisque le but est de maximiser l'erreur sur $\mathcal{S}^-$ et de la minimiser sur $\mathcal{S}^+$, les atomes les plus intéressants pour la création d'une règle logique sont donc ceux présents en haut à gauche de la figure. Ce graphique permet également de visualiser le score des atomes sous forme de droites. Les atomes ayant un score de 0, à savoir a_1 et a_0, sont ainsi situés sur la droite représentée en gris pointillé et passant par les points $(0, 0)$ et $(|\mathcal{S}^+|, |\mathcal{S}^-|)$ $(= (4, 4))$. Les atomes situés en dessous de cette droite sont donc ceux ayant un score négatif et ceux placés au-dessus sont ceux ayant un score positif. Sur ce graphique, on peut également visualiser les symétries entre les valeurs discrètes 0 et 1 d'une même variable ($erreur^+(v_i) = |\mathcal{S}^+| - erreur^+(v_{1-i})$ et $erreur^-(v_i) = |\mathcal{S}^-| - erreur^-(v_{1-i})$). Le seuil de sélection des atomes ayant été fixé à $t = 0.4$ dans cet exemple, on peut également facilement visualiser les atomes sélectionnés pour la règle optimale. Ces derniers correspondent aux atomes situés au-dessus de la droite correspondant au score de 0.4 en noire, et sont représentés en vert.

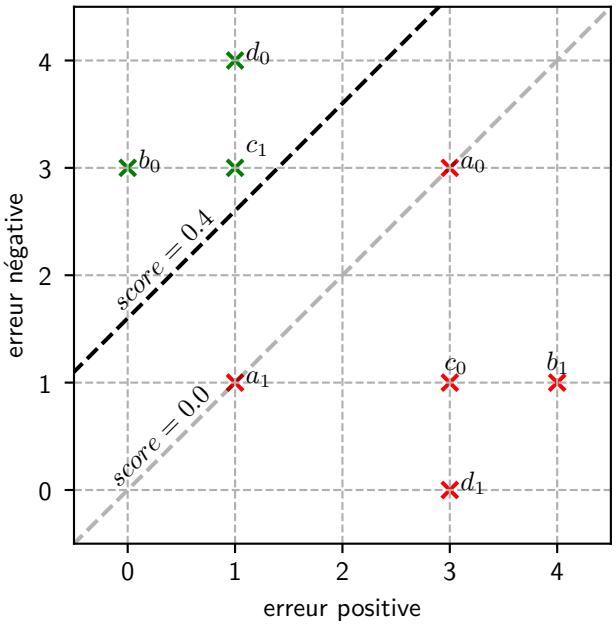

FIGURE 3.4 – Représentation graphique des erreurs positives et négatives de chaque atome.

Ainsi, la règle optimale *LOLH* de cet exemple est la règle r^* telle que $corps(r^*) = \{b_0, d_0, c_1\}$. Bien que l'atome c_1 ait un score moins élevé que b_0 et d_0, il est tout de même sélectionné par rapport au seuil t. On peut notamment vérifier, comme expliqué précédemment, que le retrait de ce dernier atome permet d'améliorer le score (normalisé) de la règle. En effet, si on considère la règle r' avec $corps(r') = \{b_0, d_0\}$, on a alors $\frac{score(r')}{|corps(r')|} = (0.75 + 0.75)/2 = 0.75$. Or $\frac{score(r^*)}{|corps(r^*)|} = (0.75 + 0.75 + 0.5)/3 = 2/3 \approx 0.67 < score(r')/|corps(r')|$.

On peut finalement remarquer qu'un score négatif pour un atome ne signifie pas forcément que cet atome est un mauvais candidat. En effet, un score négatif signifie que l'atome est majoritairement présent dans les exemples négatifs et qu'il n'est majoritairement pas présent dans les exemples positifs. Cela implique que la prédiction inverse est une bonne prédiction. Dans le cas où les variables sont binaires, il est ainsi intéressant de sélectionner l'atome ayant l'autre valeur discrète pour la même variable. Dans le cas multivalué, on pourrait également introduire l'utilisation de la négation $\neg v_i$ d'un atome logique dans une règle afin d'inverser la prédiction. Les atomes les moins intéressants sont donc ceux ayant un score proche de 0, comme c'est le cas avec a_0 et a_1. Ces atomes sont ainsi situés près de la droite en gris pointillé.

3.4 Application artificielle

Afin d'illustrer notre approche d'optimisation pour l'inférence de règles *LFIT*, nous appliquons *LOLH* sur un jeu de données généré artificiellement. Nous avons ainsi généré aléatoirement un jeu de données contenant 2000 états discrets, pour 10000 variables binaires. Ce jeu de données a été généré de manière à imiter les données d'expressions génétique *single-cell*, à partir desquelles *LOLH* a été inspiré (voir chapitre 4). La méthode mise en place pour la génération de ces données est détaillée en annexe A. Dans ce jeu de données, deux ensembles d'exemples positifs et négatifs sont donnés, avec $|\mathcal{S}^+| = 1000$ et $|\mathcal{S}^-| = 1000$, pour les 10000 variables binaires. Il y a donc 20000 atomes logiques possibles pour former les règles logiques. Nous proposons d'utiliser ces données pour illustrer l'intérêt de l'erreur de couverture d'une règle logique (définition 3.5 en page 98) appliquée à un exemple de grande taille. Nous proposons notamment une comparaison de *LOLH* avec une implémentation existante de *LFIT*. *GULA* n'étant pas assez rapide pour être exécuté sur des volumes de données d'aussi grande taille, nous utilisons l'implémentation

heuristique *PRIDE* (voire partie 3.2.2 en page 89) pour réaliser cette comparaison. Les analyses réalisées ici sont par ailleurs reproductibles à l'aide de notre implémentation en ligne [7].

3.4.1 Comparaison entre *LOLH* et *PRIDE*

Dans cette partie, nous comparons les performances de *LOLH* et *PRIDE* pour la classification de $\mathcal{S}^+$ et $\mathcal{S}^-$. Nous effectuons également la comparaison avec une règle logique obtenue de manière aléatoire, afin de mettre en évidence la difficulté du problème. Dans un premier temps, nous détaillons la manière dont les règles de *LOLH* et *PRIDE* sont obtenues, ensuite nous évaluons les performances des deux algorithmes avec une méthode de validation croisée pour la classification des données. Enfin, nous développons l'analyse en étudiant les scores (normalisés) des règles logiques ainsi que les scores des atomes.

Construction des classifieurs à partir des règles logiques

Pour appliquer *LOLH* sur cet exemple artificiel, nous utilisons l'implémentation *python* [8] de l'algorithme 4 avec en paramètre un seuil de **0.6** pour la sélection des atomes. On obtient alors une règle logique dont le corps contient 22 atomes, et que nous appelons par la suite $\mathcal{R}_{LOLH}$ (avec ainsi $|corps(\mathcal{R}_{LOLH})| = 22$). Nous devons ensuite déterminer une manière de prédire si un état discret s est un exemple positif ou négatif, en utilisant la règle $\mathcal{R}_{LOLH}$. Nous considérons ici que s est prédit comme étant un exemple positif si et seulement si l'erreur de couverture de $\mathcal{R}_{LOLH}$ sur s est inférieure ou égale à la moitié de la longueur de $\mathcal{R}_{LOLH}$:

$$prediction_{LOLH}(s) = \begin{cases} \text{“}\mathcal{S}^+\text{”}\ si\ erreurCouverture(\mathcal{R}_{LOLH}, s) \leq \frac{|corps(\mathcal{R}_{LOLH})|}{2} \\ \text{“}\mathcal{S}^-\text{”}\ sinon \end{cases}$$

L'inférence de règles avec l'algorithme *PRIDE* est effectuée via l'implémentation *pylfit* disponible en ligne [9]. Bien que l'implémentation ait été initialement prévue pour inférer un programme logique entier à partir d'un ensemble de transitions, nous utilisons seulement

7. https://github.com/smbct/LOLH.git
8. L'implémentation est disponible en ligne à l'adresse https://github.com/smbct/LOLH.git.
9. https://github.com/Tony-sama/pylfit.git

la fonction *"fit_var_val"* qui retourne directement un ensemble de règles étant donné un ensemble d'exemples positifs et un ensemble d'exemples négatifs. L'application de *PRIDE* sur $\mathcal{S}^+$ et $\mathcal{S}^-$ résulte en un ensemble de 185 règles logiques, dont les corps contiennent entre 3 et 11 atomes (la majorité des règles contenant entre 5 et 7 atomes). Nous notons $\mathcal{P}_{PRIDE}$ l'ensemble de règles obtenu, avec ainsi $|\mathcal{P}_{PRIDE}| = 185$. Afin de proposer une prédiction pour un état s en utilisant $\mathcal{P}_{PRIDE}$, on prédit un état s comme un exemple positif si et seulement s'il existe au moins une règle de $\mathcal{P}_{PRIDE}$ couvrant s :

$$prediction_{PRIDE}(s) = \begin{cases} \text{“}\mathcal{S}^+\text{”} \ si \ \exists r \in \mathcal{P}_{PRIDE} : couverture(r, s) = \top \\ \text{“}\mathcal{S}^-\text{”} \ sinon \end{cases}$$

Enfin, nous construisons une règle aléatoire $\mathcal{R}_{ALÉATOIRE}$ telle que $|corps(\mathcal{R}_{ALÉATOIRE})| = |corps(\mathcal{R}_{LOLH})|$. Cette règle est construite en choisissant de manière totalement aléatoire un ensemble d'atomes logiques parmi tous les atomes possibles à partir des données de l'instance. La prédiction d'un état s pour de ce modèle est calculée de la même manière que pour la règle $\mathcal{R}_{LOLH}$.

Validation croisée sur $\mathcal{S}^+$ et $\mathcal{S}^-$

Afin d'illustrer l'efficacité de la méthode d'inférence *LOLH* par rapport à *PRIDE*, nous effectuons la comparaison à partir d'une validation croisée. En effet, l'évaluation d'un modèle sur le jeu de données à partir duquel il a été inféré (que l'on appelle généralement le jeu de données d'entraînement) n'est généralement pas une bonne idée car cela ne reflète pas les performances de généralisation du modèle. Par exemple, les performances de l'algorithme *PRIDE* sur ses données d'entraînement sont toujours optimales étant donné que les règles sont inférées de manière à ne produire aucune erreur sur ces dernières. Pour comparer *LOLH*, *PRIDE* et le modèle aléatoire, nous procédons donc à une validation croisée *k-fold* avec $k = 10$. Cette méthode consiste à créer k sous-ensembles des données originales. Les modèles (c'est-à-dire les ensembles de règles) sont ensuite inférés sur un des k sous-ensemble, et ils sont évalués sur les données restantes (Liu et al., 2008). Le processus est répété jusqu'à ce que les modèles aient été entraînés sur l'ensemble des k sous-ensembles.

Pour chacune des 10 validations, les prédictions des différents algorithmes sont résumées en deux valeurs entre 0 et 1 sur les exemples positifs et négatifs respectivement,

correspondant au pourcentage d'états prédits comme positifs [10]. De bonnes performances sont ainsi atteintes si les valeurs obtenues sont proches de 1 sur les exemples positifs et de 0 sur les exemples négatifs. Le tableau 3.4 présente la moyenne et la variance des 10 taux de prédictions positives obtenues par *PRIDE*, *LOLH* et par la règle aléatoire sur $\mathcal{S}^+$ et $\mathcal{S}^-$. On remarque que la stratégie employée par *LOLH* permet une classification parfaite sur les données d'évaluation. Par ailleurs, *PRIDE* obtient une bonne moyenne sur les exemples positifs. Cependant, les prédictions de *PRIDE* conduisent à beaucoup de faux positifs, ce qui donne une moyenne élevée, donc mauvaise, sur les exemples négatifs. Cela semble correspondre à son objectif qui consiste à inférer des règles générales, risquant ainsi la couverture d'exemples négatifs. Comme on peut s'y attendre, les performances du modèle aléatoire sont moins bonnes, puisque le taux de prédictions positives moyen de ce modèle est assez faible sur $\mathcal{S}^+$ et assez élevé sur $\mathcal{S}^-$. Par ailleurs, dans les trois cas, les prédictions semblent stables sur les données d'évaluation, puisque la variance des taux de prédictions positives est faible.

algorithme	$\mathcal{S}^+$		$\mathcal{S}^-$	
	moyenne	variance	moyenne	variance
LOLH	9.99×10^{-1}	1.11×10^{-7}	1.11×10^{-4}	1.11×10^{-7}
PRIDE	7.95×10^{-1}	8.57×10^{-4}	4.17×10^{-1}	3.14×10^{-3}
ALÉATOIRE	3.72×10^{-1}	2.51×10^{-2}	4.42×10^{-1}	3.45×10^{-2}

TABLE 3.4 – Moyennes et variances des taux de prédictions positives sur les 10 ensembles de données d'évaluation pour les différents algorithmes testés.

Analyses des règles *LOLH* et *PRIDE*

Afin d'aller plus loin dans la comparaison entre *LOLH* et *PRIDE*, on peut regarder plus en détail le comportement des règles selon l'erreur de couverture sur les exemples positifs et sur les exemples négatifs. Dans cette partie, nous inférons un modèle $\mathcal{R}_{LOLH}$ avec un seuil de 0.6 et un modèle $\mathcal{P}_{PRIDE}$ en utilisant l'implémentation en ligne. Dans un premier temps, la comparaison entre *LOLH* et *PRIDE* peut être effectuée en utilisant le score de $\mathcal{R}_{LOLH}$ et des règles $\mathcal{P}_{PRIDE}$. Nous décomposons ici le score des règles en une erreur positive normalisée ($erreur^+(r) = \left(\sum_{v_i \in corps(r)} erreur^+(v_i)\right) / \left(|\mathcal{S}^+| * |corps(r)|\right)$) et une

10. Par exemple, une valeur de 0.8 sur les ensembles $\mathcal{S}^+$ indiquent que 80% des exemples positifs dans les données d'évaluation ont été prédits comme positifs.

erreur négative normalisée ($erreur^-(r) = \left(\sum_{v_i \in corps(r)} erreur^-(v_i)\right) / (|\mathcal{S}^-| * |corps(r)|)$), dans le but de comparer des règles de longueur différente sur les exemples positifs et sur les exemples négatifs séparément. En reprenant la définition 3.6 du score d'une règle en page 99, on a alors $erreur^+(r) - erreur^-(r) = score(r)/|corps(r)|$. La figure 3.5 permet de comparer les erreurs positives et négatives des règles inférées par *PRIDE* avec l'erreur positive et l'erreur négative de la règle inférée par *LOLH*. La diagonale en noir représente les règles pour lesquelles le score est égal à 0. On remarque que la règle inférée par *LOLH* (en orange) est meilleure que presque toutes les règles inférées par *PRIDE* car son erreur positive est plus faible et son erreur négative est plus élevée respectivement que celles des règles de *PRIDE*. De plus, les points correspondant aux règles de *PRIDE* sont relativement peu éloignés de la droite noire, ce qui indique que *PRIDE* prend peu en considération l'erreur de couverture introduite dans la définition 3.5 en page 98. On remarque par ailleurs que les règles de *PRIDE* qui ont le plus d'atomes dans leur corps sont aussi celles qui ont les plus petites erreurs (à la fois sur les exemples positifs et négatifs).

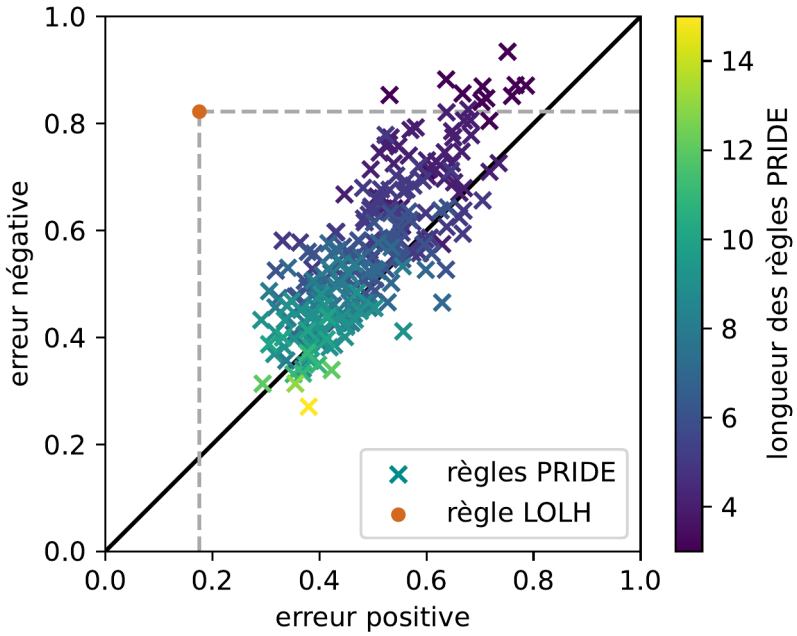

FIGURE 3.5 – Visualisation des erreurs positives et négatives des règles inférées par *PRIDE* (en bleu et jaune) et de la règle inférée par *LOLH* (en orange).

Pour étudier plus en détails les règles inférées par *PRIDE* et *LOLH*, on peut visualiser l'erreur de couverture d'une règle donnée sur l'ensemble des données à partir de deux histogrammes. La figure 3.6 permet de visualiser l'erreur de couverture de $\mathcal{R}_{LOLH}$ à l'aide d'un histogramme sur les exemples positifs (en vert), et d'un histogramme sur les exemples négatifs (en rouge). Ainsi, pour chaque valeur d'erreur possible, l'histogramme représente la proportion des états sur lesquels la règle obtient cette erreur. On remarque que la règle $\mathcal{R}_{LOLH}$ permet de séparer efficacement $\mathcal{S}^+$ et $\mathcal{S}^-$ puisque les histogrammes ne se chevauchent pas. Puisqu'il n'existe pas d'exemples positifs et négatifs ayant la même erreur de couverture, cette règle permet alors de classifier parfaitement les exemples (même si cela ne donne pas d'indications sur les performances du modèle en validation croisée).

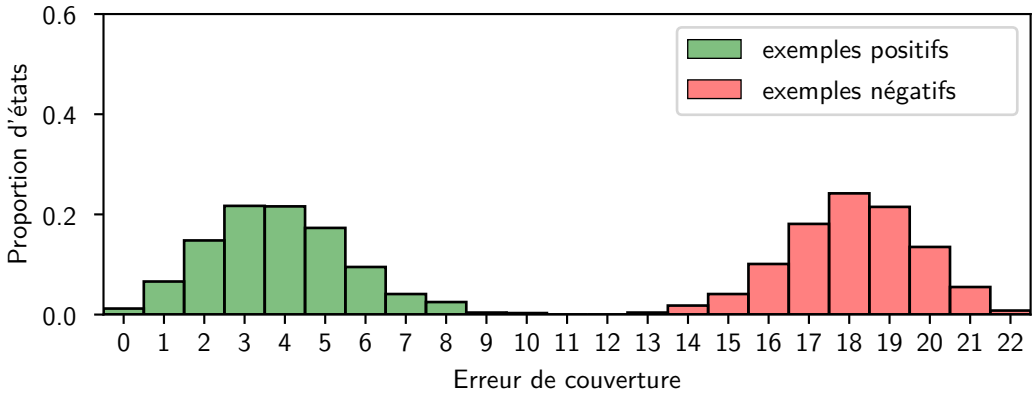

FIGURE 3.6 – Histogrammes de l'erreur de couverture de $\mathcal{R}_{LOLH}$ sur les exemples positifs (en vert) et sur les exemples négatifs (en rouge).

De manière similaire, on peut tenter de visualiser les histogrammes des règles dans $\mathcal{P}_{PRIDE}$. Puisqu'il est difficile de visualiser à la fois les 185 règles inférées, nous nous concentrons sur certaines d'entre elles. La figure 3.7 représente les histogrammes de deux règles : une des règles les plus longues, contenant 11 atomes, et une des meilleures règles au sens du score normalisé, contenant 3 atomes. On remarque que les deux histogrammes de la règle contenant 11 atomes (figure de gauche) sont complètement superposés, ce qui indique que cette règle ne permet pas de généraliser la classification sur $\mathcal{S}^+$ et $\mathcal{S}^-$. La règle contenant 3 atomes (à droite) permet de mieux séparer les exemples positifs des négatifs, ce qui justifie son score normalisé intéressant. Cependant, on remarque tout de même un chevauchement des deux histogrammes laissant penser que la généralisation n'est pas idéale en comparaison avec celle de $\mathcal{R}_{LOLH}$.

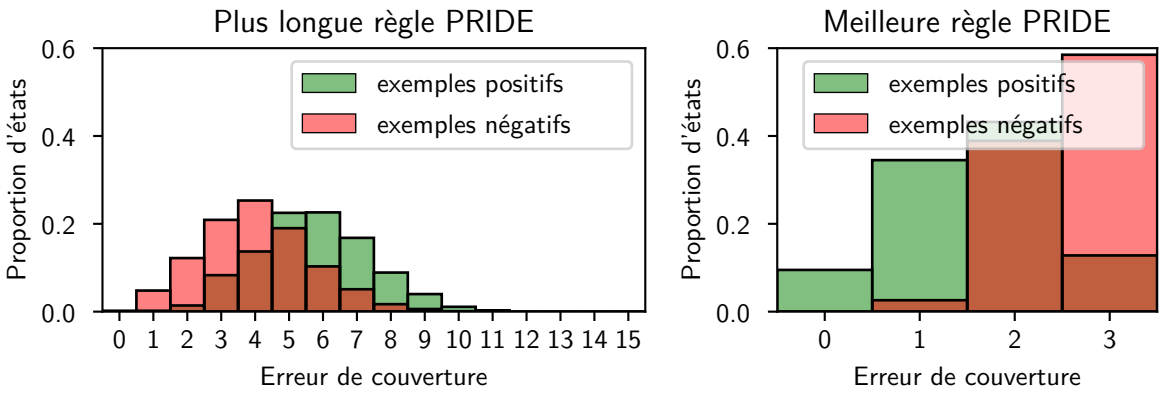

FIGURE 3.7 – Histogrammes de l'erreur de couverture de la plus longue règle de *PRIDE* à gauche (avec 11 atomes dans le corps), et de la meilleure règle de $\mathcal{P}_{PRIDE}$ selon le score normalisé à droite (avec 3 atomes). L'histogramme sur les exemples positifs est en vert et celui sur les exemples négatifs en rouge.

Alternativement à l'analyse individuelle des 185 règles de *PRIDE*, on peut étudier visuellement les erreurs positives et négatives des atomes sélectionnés à partir la définition 3.7 en page 100, comme cela a été déjà proposé dans l'exemple jouet. La figure 3.8 représente ainsi les erreurs des atomes sélectionnés par *PRIDE* en rouge, celles des atomes sélectionnés par *LOLH* en vert, ainsi que les atomes non utilisés en bleu. La droite pointillée représente les atomes dont le score est 0 (c'est-à-dire quand l'erreur positive et l'erreur négative sont égales) et la droite continue représente le score de 0.6 utilisé ici comme seuil pour *LOLH*. On remarque sur cette figure l'effet de l'optimisation du score par *LOLH* puisque les atomes sélectionnés sont ceux présents en vert, en haut à gauche du graphique, minimisant ainsi l'erreur positive et maximisant l'erreur négative. Au contraire, les atomes sélectionnés par *PRIDE* ne sont pas les plus avantageux du point de vue des erreurs et de leur score, car ceux-ci sont regroupés autour de la droite pointillée, et certains d'entre eux ont un score négatif. Cela confirme les observations précédentes concernant les scores des règles et les histogrammes. Les atomes sélectionnés dans les règles de $\mathcal{P}_{PRIDE}$ présentent cependant une spécificité non visible sur la figure 3.8. En effet, les atomes présents dans le corps des règles inférées par *PRIDE* appartiennent aux 25 premières variables du jeu de données artificiel. Cela indique que *PRIDE* n'est pas en mesure d'exploiter l'ensemble des 10000 variables pour inférer les règles. Une des raisons pouvant expliquer cela est le fait que les atomes sont ajoutés de manière itérative dans les règles de *PRIDE*, en itérant de manière croissante sur les indices des variables. Il semble alors être assez immédiat de trouver des atomes permettant de ne couvrir aucun exemple négatif, ce qui explique le fait que très peu de variables sont considérées dans l'inférence des règles.

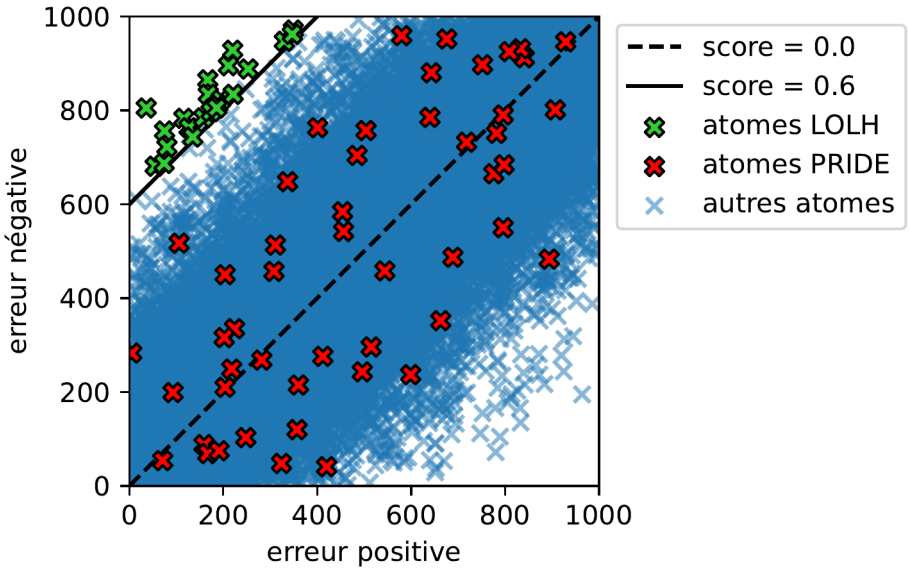

FIGURE 3.8 – Visualisation des erreurs positives et négatives des atomes. Les atomes sélectionnés dans les règles par *PRIDE* et *LOLH* sont représentés en rouge et vert respectivement.

3.4.2 Comparaison de différentes règles logiques

Afin de bien comprendre le lien entre les erreurs positives et les erreurs négatives des atomes logiques et le score d'une règle logique, on peut comparer plusieurs règles différentes construites selon différents critères de sélection des atomes. Nous créons ainsi 4 règles logiques contenant chacune 10 atomes logiques, à partir de 4 critères différents. Les trois premières règles sont construites en sélectionnant aléatoirement 10 atomes dont les scores sont compris entre deux bornes. Une première règle nommée "règle sous-optimale" est obtenue en sélectionnant des atomes dont les scores sont compris entre 0.3 et 0.4, permettant d'obtenir une règle légèrement moins bonne que la règle optimale. La deuxième règle, appelée "règle inverse", est obtenue en sélectionnant les atomes dont le score est compris entre -0.8 et -0.7, et permet ainsi de classer "inversement" $\mathcal{S}^+$ et $\mathcal{S}^-$ La troisième règle, appelée "règle inappropriée", consiste à sélectionner des atomes dont les scores sont compris entre -0.05 et 0.05. Enfin une dernière règle appelée "règle aléatoire" est construite en sélectionnant aléatoirement 10 atomes parmi tous les atomes possibles. Les erreurs positives et négatives des atomes sélectionnés pour ces 4 règles peuvent être visualisés sur la figure 3.9.

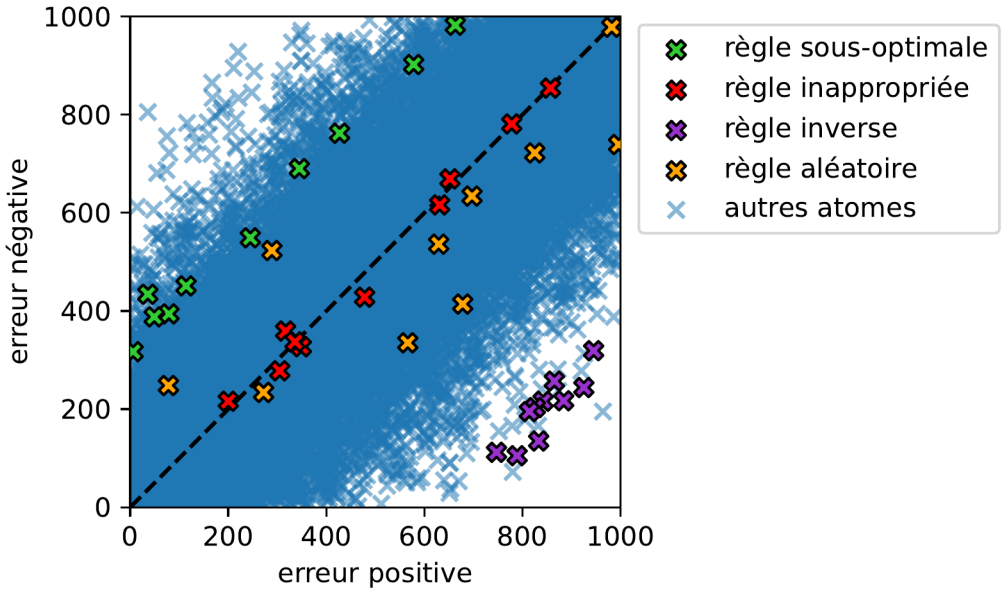

FIGURE 3.9 – Erreurs positives et négatives des atomes sélectionnés pour les différentes règles.

La figure 3.10 représente les histogrammes des 4 règles décrites précédemment. On remarque que la règle "sous-optimale" ne permet pas de séparer complètement les exemples positifs et négatifs, ce qui confirme que cette règle est moins intéressante que $\mathcal{R}_{LOLH}$, dont les histogrammes sont représentés sur la figure 3.6. La règle "inverse" permet au contraire une séparation parfaite de $\mathcal{S}^+$ et $\mathcal{S}^-$, mais cette fois-ci avec les exemples négatifs ayant une erreur de couverture faible et les exemples positifs ayant une erreur de couverture élevée. Cela justifie le fait que les atomes ayant des scores négatifs proches de 1 sont tout de même intéressants pour la classification, comme cela a été mentionné à la fin de la partie 3.3.3. Enfin, la règle "inappropriée" et la règle "aléatoire" sont similaires, avec des histogrammes qui se chevauchent totalement. Ces règles montrent qu'il n'est pas intéressant de sélectionner les atomes ayant un score proche de 0, ou encore d'utiliser ensemble des atomes ayant des scores positifs et des scores négatifs.

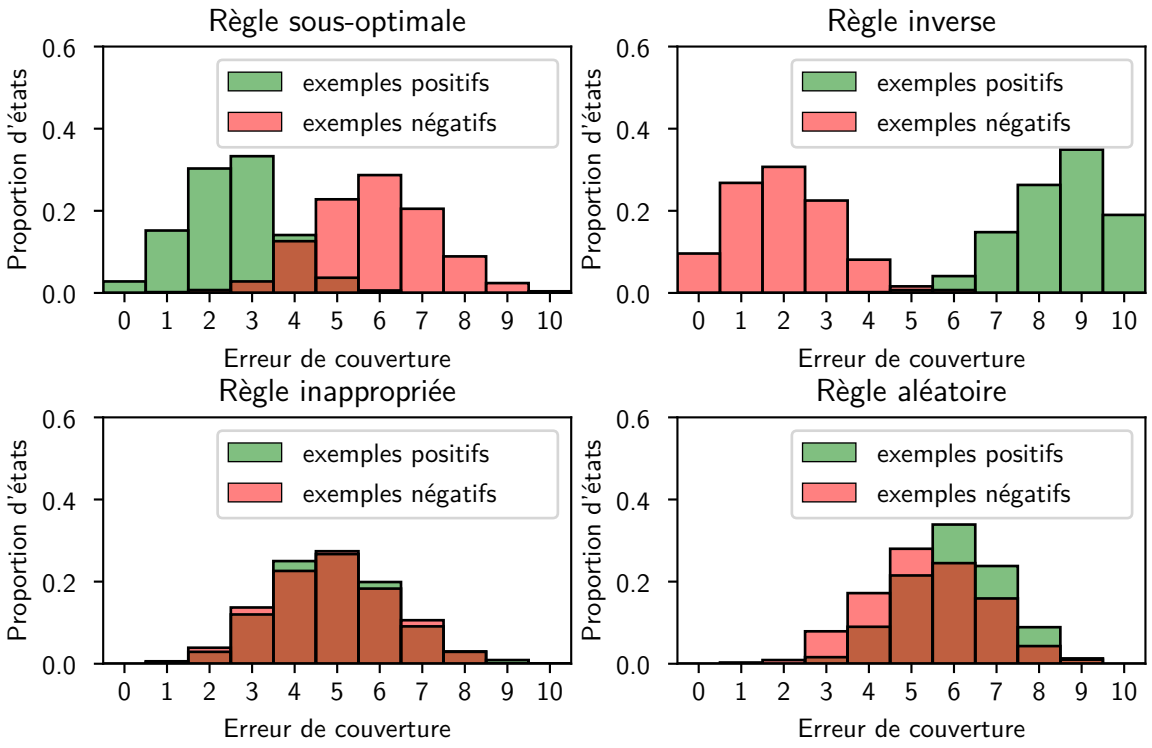

FIGURE 3.10 – Comparaison des histogrammes de l'erreur de couverture des différentes règles.

3.4.3 Optimisation biobjectif pour l'inférence

On pouvait remarquer précédemment que la formulation initiale du problème d'optimisation des règles à partir de l'erreur de couverture (expression 3.1 en page 99) conduit à considérer deux objectifs à optimiser : la minimisation d'un score sur les exemples positifs et la maximisation d'un score sur les exemples négatifs. Bien que dans le contexte de l'inférence de règles pour la classification de $\mathcal{S}^+$ et $\mathcal{S}^-$ il soit difficile de choisir d'optimiser en priorité sur les exemples positifs ou sur les exemples négatifs, il est tout de même intéressant d'envisager ce problème sous l'angle de l'optimisation biobjectif. La résolution d'un problème d'optimisation à deux objectifs ne conduit alors pas à une unique solution optimale, mais à un ensemble de solutions (compromis) optimales et incomparables entre elles. Une solution (règle) est alors considérée optimale s'il n'existe pas de règle meilleure à la fois sur le score positif et sur le score négatif.

En fixant à l'avance le nombre d'atomes des règles logiques, on peut alors calculer différentes règles (compromis) optimales. L'ensemble des solutions optimales d'un problème

multiobjectif, c'est-à-dire le front de *pareto*, est en général difficile à obtenir, même pour des problèmes dont leur équivalent mono-objectif est polynomial (Ruhe, 1988). On peut cependant calculer un sous-ensemble de ces solutions, appelées *solutions supportées*, en optimisant une somme pondérée des deux objectifs initiaux, et en se ramenant ainsi à l'optimisation d'un unique objectif. Nous mettons cela en pratique sur les données artificielles en fixant la longueur des règles à $k = 10$ atomes. On peut alors calculer l'ensemble des "règles supportées" en utilisant l'algorithme décrit dans (Aneja et al., 1979), et en résolvant les problèmes mono-objectifs avec *LOLH*. La figure 3.11 représente ainsi les scores positifs et négatifs des règles obtenues par pondération.

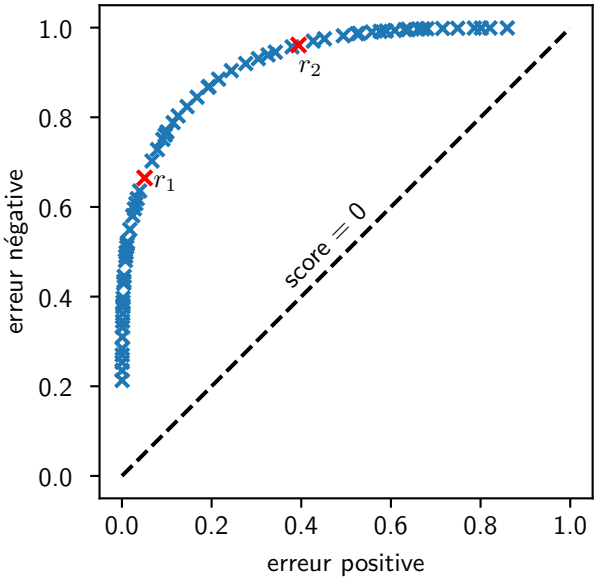

Figure 3.11 – Représentation des scores positifs et négatifs des règles supportées pour l'optimisation biobjectif.

On peut alors visualiser les histogrammes de plusieurs règles pour comparer différentes pondérations des scores positifs et négatifs. On se concentre en particulier sur les règles r_1 et r_2 représentées en rouge sur la figure 3.11, et dont les histogrammes sont donnés sur la figure 3.12. On remarque que r_1 optimise en priorité le score positif car son score positif est meilleur que celui de r_2, et son score négatif est moins bon. Cela peut se vérifier sur les histogrammes de la figure 3.12 puisque les exemples positifs ont presque tous une erreur de couverture nulle pour r_1 alors que certains exemples négatifs ont une erreur de couverture négative qui n'est pas très élevée. Cette tendance est inversée pour la règle r_2.

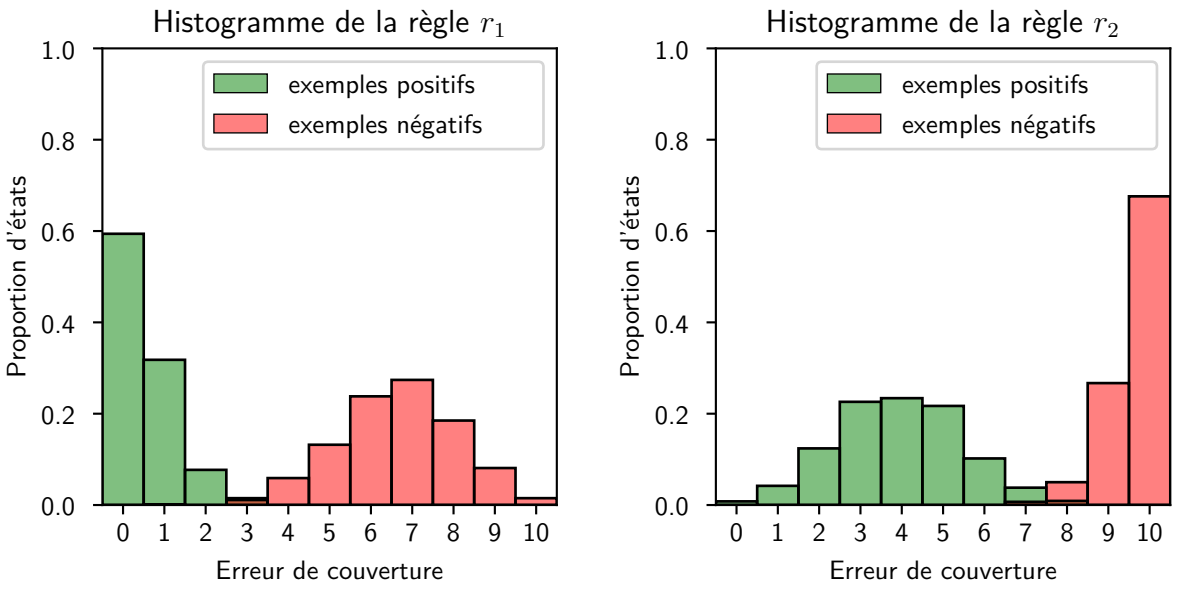

FIGURE 3.12 – Histogrammes des règles r_1 et r_2 représentées dans la figure 3.11.

3.5 Discussions

Nous avons vu dans ce chapitre que plusieurs stratégies ont été proposées dans la littérature pour l'inférence des modèles qualitatifs. La formulation de problèmes de satisfaction de contraintes ou encore de problèmes d'optimisation a ainsi été employée, avec l'utilisation de plusieurs données d'inférence telles que des graphes d'influence, des propriétés dynamiques ou encore des données d'expression génétique. Les approches *LFIT* se révèlent alors intéressantes car celles-ci permettent d'exploiter au maximum les données expérimentales disponibles, réduisant alors le risque d'*overfitting* par rapport aux connaissances déjà acquises sur le système considéré. Un des intérêts de *LFIT* réside également dans son schéma d'apprentissage, consistant à transformer le problème d'inférence de la dynamique en un ensemble de problèmes de classification, plus classiques à traiter. L'approche d'optimisation présentée dans ce chapitre vise ainsi à s'abstraire totalement du contexte dynamique, et à se concentrer sur l'inférence du modèle de classification le plus efficace et robuste possible. Notre méthode, *LOLH*, se base ainsi sur la formulation d'un problème d'optimisation dans un cadre de modélisation logique afin de correctement prendre en compte le bruit des données. Celle-ci est rapide à exécuter, et nous avons notamment montré son intérêt pour l'application sur des données bruitées en comparaison avec d'autres implémentations existantes de *LFIT*.

Plusieurs éléments restent alors à considérer concernant l'inférence des règles avec *LOLH*. Tout d'abord, la méthode présentée ici ne permet d'inférer qu'une seule règle logique. Bien que cette approche permette de classifier parfaitement les données dans l'exemple artificiel proposé, nous pourrons remarquer dans le chapitre 4 que la difficulté d'une instance de classification peut varier significativement sur des données réelles, en fonction de la variable considérée pour l'inférence des règles. Par ailleurs, un des intérêts de l'approche *LFIT* réside dans la possibilité d'interpréter formellement le modèle résultant. L'inférence de plusieurs règles logiques peut alors être un avantage dans ce contexte, et plusieurs pistes ont été envisagées comme la formulation d'un problème d'optimisation biobjectif ou encore l'inférence de règles plus spécifiques sur les exemples positifs.

L'aspect dynamique du modèle, qui a peu été abordé dans le cadre de notre approche d'optimisation, nécessite également d'être développé davantage pour ce nouvel algorithme. Dans un premier temps, il peut être intéressant de tester plusieurs méthodes de prédiction à partir d'un score de couverture. En effet, la méthode de validation présentée dans la dernière partie de ce chapitre consiste à comparer l'erreur de couverture d'une règle avec la moitié de sa longueur. On peut imaginer que cette stratégie peut poser des difficultés sur certaines instances, notamment où les histogrammes des erreurs de couverture se chevauchent ou sont décalés par rapport au milieu de l'histogramme. On peut également se poser la question de la sémantique à utiliser avec un tel modèle. Il peut sembler pertinent de se tourner vers une sémantique **synchrone** dans le cadre de l'apprentissage automatique avec *LOLH*. En effet, une règle logique inférée par *LOLH* a pour but de proposer une prédiction sur l'évolution d'une des variables du système. Or, en sémantique asynchrone ou asynchrone généralisée, le modèle est autorisé à n'effectuer aucune évolution sur certaines variables entre deux transitions. Cela revient alors à n'effectuer aucune prédiction sur une des variables, alors que d'autres variables ont la possibilité d'évoluer. On peut ainsi interpréter cet aspect comme une "désynchronisation" du modèle puisque les variables qui ne sont pas mises à jour dans le nouvel état peuvent être considérées comme non prédites, c'est-à-dire que le modèle n'est pas capable de justifier leur valeur. L'utilisation d'une sémantique synchrone permet alors de simuler uniquement des états qui sont vraisemblables par rapport ce qui a été inféré à partir des données. Dans cette logique, il est par ailleurs également nécessaire de "bloquer" l'évolution du modèle quand aucune prédiction n'est possible pour au moins une des variables, ce qui n'est par ailleurs pas

proposé dans le cadre des réseaux d'automates synchrones. On pourrait aussi considérer la prédiction non déterministe en sémantique synchrone, ce qui semble particulièrement intéressant pour les données *single-cell*. Or, en synchrone, avec une seule règle, il n'y a pas forcément de non-déterminisme.

Un dernier aspect intéressant à considérer est la vérification des propriétés dynamiques abordée dans le chapitre 2, en particulier avec l'utilisation du formalisme de la programmation logique. Comme nous l'avons mentionné au début de ce chapitre, il existe une forte proximité entre les programmes logiques inférés par *LFIT* et les réseaux d'automates. On peut alors imaginer de réutiliser certaines des méthodes présentées dans le chapitre 2, notamment le *Bounded Model Checking*, afin de vérifier facilement des propriétés dynamiques dans des programmes logiques de grande taille.

Il est par ailleurs important d'évaluer l'algorithme *LOLH* sur des données réelles afin d'en déterminer son intérêt pour l'étude des systèmes biologiques. Dans le chapitre 4, nous nous concentrons sur les données d'expression génétique issues des technologies de séquençage *single-cell*, qui ont notamment motivé la mise en place de *LOLH*. Ces données entrent dans le cadre des hypothèses formulées dans ce chapitre car elles sont de grande dimension, elles apportent dans une certaine mesure des informations dynamiques, et elles sont souvent disponibles en grande quantité. L'inférence de la dynamique étant un problème complexe, nous le traitons par étape dans le chapitre 4 en formulant plusieurs problèmes d'inférence différents à partir d'un même jeu de données. Nous proposons ainsi plusieurs méthodes d'interprétation des modèles résultants afin d'évaluer au mieux les capacités de *LOLH* dans un contexte biologique concret.

Chapitre 4

Application en séquençage single-cell

Résumé

Dans ce chapitre, nous appliquons l'algorithme *LOLH* mis au point dans le chapitre précédent sur des données d'expression génétique obtenues à partir des technologies de séquençage *single-cell*. Nous présentons dans un premier temps ces technologies et nous motivons le choix de ces données pour notre approche de modélisation qualitative. Ensuite, nous appliquons *LOLH* de deux manières différentes. Dans un premier temps, nous l'appliquons pour l'inférence d'un réseau de co-expression, dont l'exploitation permet de calculer des propriétés biologiques. Dans un deuxième temps, nous proposons une approche pour l'inférence d'un réseau dynamique. Nous montrons notamment que ce réseau capture des propriétés différentes par rapport à l'inférence de la co-expression.

4.1 Séquençage *single-cell* (ARN)

Le séquençage *single-cell* désigne un ensemble de technologies apparues récemment qui permettent de séquencer individuellement un grand nombre de cellules à partir d'un unique échantillon biologique. Bien que ces technologies aient été développées pour plusieurs types d'analyse, nous nous concentrons ici sur le séquençage des molécules d'ARN. En comparaison, les technologies de séquençage non *single-cell* permettent de séquencer les molécules d'ARN provenant des différentes cellules d'un échantillon de manière groupée seulement. Ce type de séquençage résulte donc en un profil génétique "moyenné" des

cellules étudiées. Le *single-cell* améliore ainsi la résolution avec laquelle il est possible d'étudier l'expression génétique d'un système biologique puisqu'il renseigne sur l'activité de dizaines de milliers de gènes, pour plusieurs milliers de cellules à partir d'un seul séquençage (Potter, 2018). Cette avancée technologique a notamment plusieurs fois été mise en avant en tant que *Méthode de l'année* dans le journal *Nature method* (« Method of the Year 2013 », 2014 ; « Method of the Year 2019 : Single-cell multimodal omics », 2020). Bien que ces données soient généralement bruitées (il est en effet difficile de détecter les ARN lorsque peu de molécules sont présentes) et que les technologies associées sont coûteuses, leur précision les a rendues couramment utilisées en recherche biomédicale. Au-delà de l'intérêt que représente une telle quantité de données pour l'inférence de modèles biologiques, un aspect très intéressant des données *single-cell* est l'interprétation dynamique qu'il est possible d'en tirer. C'est cet aspect, détaillé plus tard dans ce chapitre, qui nous motive principalement pour l'application de *LOLH* sur ces données.

4.1.1 Principe général

Les technologies de séquençage *single-cell* reposent sur la possibilité d'isoler les différentes cellules d'un échantillon biologique, et d'en extraire le contenu génétique. Bien que ces technologies améliorent grandement la résolution avec laquelle il est possible d'étudier l'expression génétique, les difficultés rencontrées pour leur mise en pratique ont un impact sur la manière dont les données sont analysées. Nous abordons donc les différentes étapes de leur réalisation afin de mieux comprendre les traitements réalisés sur les données associées à ces technologies. La première étape consiste à dissocier les cellules du tissu et à les isoler individuellement. Les méthodes les plus populaires pour l'isolation des cellules sont celles basées sur l'utilisation de micro-gouttelettes avec l'aide de dispositifs microfluidiques. Ce type de technique, schématisé sur la figure 4.1, consiste à capturer les cellules dans des gouttelettes entourées d'huile. D'autres méthodes ont été proposées pour l'isolation, reposant notamment sur l'utilisation de plaques microtitres.

Lorsque les cellules sont isolées, plusieurs étapes sont réalisées avant d'effectuer le séquençage. Dans un premier temps, les cellules sont détériorées afin de n'en garder que l'ARN messager. Une séquence ARN faisant office de code-barre est alors ajoutée aux brins d'ARN pour permettre l'identification de la cellule d'origine après le séquençage. La transcription inverse est ensuite effectuée pour transformer les brins d'ARN en brins d'ADN. Puisque la quantité de molécules contenues dans une unique cellule est trop

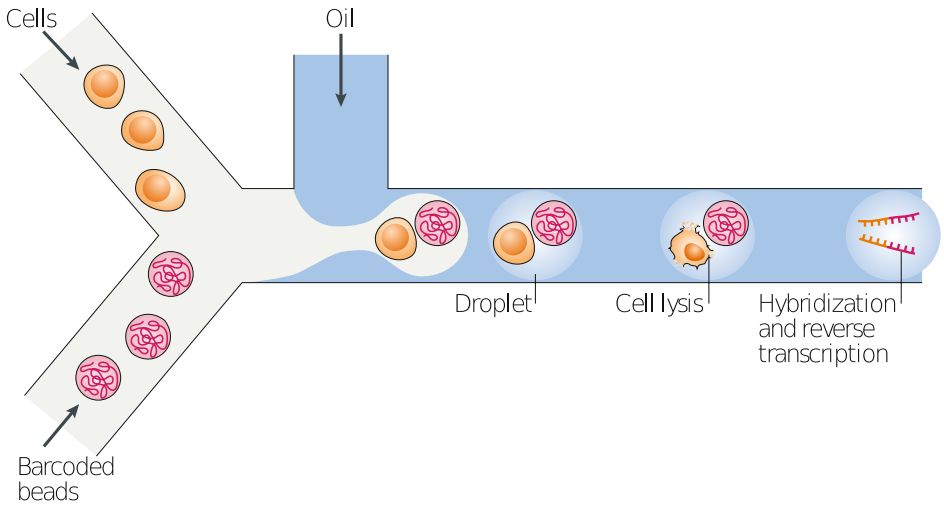

FIGURE 4.1 – Illustration des techniques d'isolation de cellules à base de gouttelettes. Figure reprise de (POTTER, 2018).

faible pour être détectée par un séquenceur, une étape d'amplification est réalisée (par exemple par *PCR* ou amplification *in-vitro*) afin d'obtenir plusieurs copies des molécules. Les brins d'ADN obtenus sont enfin séquencés à l'aide d'un séquenceur ADN haut-débit, et les séquences obtenues peuvent être alignées avec un génome de référence.

Difficultés des données *single-cell*

Les données *single-cell* prennent généralement la forme d'une matrice dont les colonnes correspondent aux cellules et dont les lignes correspondent aux gènes. Les valeurs de cette matrice correspondent ainsi au nombre d'occurrences de l'ARN messager détecté dans une cellule, transcrit à partir du gène correspondant. Plusieurs éléments importants sont à noter par rapport aux données finales. Bien que les données soient discrètes (nombre de séquences détectées correspondant à un gène donné), il n'est pas envisageable de les utiliser directement comme telles dans un formalisme de modélisation qualitatif. Tout d'abord, la transcription inverse et l'amplification des brins d'ARN comportent des biais qui font varier le nombre de molécules détectées d'une cellule à l'autre (profondeur de séquençage). Des méthodes de prétraitement sont donc nécessaires afin de corriger ces biais. De plus, les données des gènes peuvent parfois présenter beaucoup de valeurs discrètes, ce qui complique l'utilisation de formalismes qualitatifs comme *LFIT*, où l'on considère que les variables possèdent peu de valeurs qualitatives.

D'autres difficultés se présentent également à partir de la méthode expérimentale. Le séquençage d'un faible nombre de molécules est techniquement difficile, et les méthodes d'amplification sont généralement imparfaites. Il faut donc considérer que seulement les gènes les plus exprimés (c'est-à-dire les plus transcrits) sont détectés dans les cellules (détection de 10 à 20 % des molécules d'ARN présentes). Une matrice de données *single-cell* est donc une matrice creuse, c'est-à-dire contenant une majorité de 0, et les gènes non détectés mais pourtant bien présents dans les cellules sont désignés en tant que *dropout event*. Les dispositifs d'isolation à base de gouttelettes sont également imparfaits, et certains codes-barres dans les données sont donc associés au contenu de deux cellules (appelé "*doublet*"), et doivent être retirés pour ne pas biaiser l'analyse.

Aspect dynamique dans les données *single-cell*

Comme nous l'avons déjà expliqué, la modélisation qualitative qui nous intéresse dans cette thèse a pour but de décrire le comportement dynamique d'un système, et nous conduit donc à rechercher des données présentant un aspect temporel. Or, les données *single-cell* peuvent au premier abord sembler inadaptées en ce sens car elles offrent uniquement une vision statique/instantanée du système : toutes les cellules sont séquencées au même instant t. Cependant, une autre manière d'interpréter les données consiste à supposer que les cellules sont toutes issues d'un même processus d'évolution (par exemple un processus de différentiation dans un contexte développemental), mais que celles-ci ne sont pas observées dans le même état d'évolution. Cette hypothèse encourage alors une interprétation dynamique des données, bien que des relations temporelles ne soient pas explicitement connues entre les cellules. On peut notamment citer différentes méthodes de modélisation mises en place pour l'aspect dynamique, telles que les méthodes pseudo-temporelles (*pseudotime*) visant à réordonner les cellules sur un axe temporel reconstruit à partir des données, ou encore les méthodes *velocity* exploitant des informations supplémentaires sur les séquences ARN détectées afin d'inférer de l'information dynamique. C'est donc dans ce type d'approche que nous visons l'application de *LOLH* sur les données *single-cell*, et nous aborderons plus en détails par la suite certains de ces travaux.

4.1.2 Méthodes pour l'analyse des données *single-cell*

Les technologies de séquençage *single-cell* étant sensiblement différentes des technologiques de séquençage plus classiques, de nombreux nouveaux outils de calcul ont été

développés afin d'exploiter au mieux les informations capturées par ces techniques. Plusieurs outils complets implémentant toutes les étapes de traitement des données ont notamment été proposés tels que les packages R *Seurat*[1] (Hao et al., 2021) et *Monocle*[2] (Cao et al., 2019), ainsi que le package python *Scanpy*[3] (Wolf et al., 2018). Dans cette partie, nous passons en revue les méthodes principales implémentées dans ces outils pour l'analyse des données *single-cell*. Nous abordons dans un premier temps les méthodes de prétraitement, permettant de corriger certains biais expérimentaux dans le but de rendre les données exploitables. La deuxième sous-partie concerne les méthodes d'exploitation, permettant d'extraire des informations telles que les types de cellules, ou encore les gènes différemment exprimés. Certaines de ces méthodes seront utilisées par la suite pour l'évaluation de *LOLH* sur les données utilisées. Enfin, nous présentons les méthodes de modélisation dynamiques ainsi que les méthodes pour l'inférence de réseaux génétiques et de modèles qualitatifs développées pour les données *single-cell*.

Prétraitement

Les difficultés expérimentales se présentant dans le séquençage *single-cell* conduisent à la présence de bruit et de plusieurs biais dans les données collectées. Pour cette raison, plusieurs étapes de prétraitement sont nécessaires avant de pouvoir exploiter les données. La première étape, appelée contrôle de qualité, consiste à vérifier plusieurs indicateurs comme le nombre de molécules, le nombre de gènes, ainsi que le pourcentage de gènes mitochondriaux détectés dans chaque cellule. Ces vérifications permettent notamment d'exclure les *doublets*, ou encore les cellules en fin de vie (Luecken et al., 2019). Un autre élément important à prendre en compte est la variabilité du nombre de molécules détectées entre chaque cellule. En effet, les différentes étapes expérimentales ajoutent des biais qui rendent la détection plus ou moins efficace pour chaque cellule. Une étape de normalisation vise donc à corriger cet élément pour rendre les cellules directement comparables en terme d'expression des gènes. Plusieurs méthodes ont été proposées pour cette étape, une des plus utilisées consistant simplement à normaliser les valeurs d'expression par le nombre total de molécules détectées dans une cellule. Des méthodes plus avancées ont aussi été proposées, notamment la méthode *scTransform* (Hafemeister et al., 2019) basée sur l'utilisation d'une loi binomiale négative régularisée, et implémentée dans l'outil

1. *Seurat* : https://satijalab.org/seurat/.
2. *Monocle* 3 : https://cole-trapnell-lab.github.io/monocle3/
3. *Scanpy* : https://scanpy.readthedocs.io/en/stable/.

Seurat. Il est également important de noter qu'une fois les données normalisées, elles ne sont pas forcément sous forme discrète. Des biais additionnels peuvent enfin être adressés via différentes méthodes lorsque plusieurs jeux de données doivent être analysés ensemble pour comparaison. Par exemple, des cellules obtenues à partir de conditions expérimentales différentes ne sont pas directement comparables à cause de la sensibilité des données de séquençage. Des méthodes de correction de lots (*batch correction*) sont donc nécessaires pour corriger ceci. De façon intéressante, des méthodes basées sur le *deep learning* ont été proposées pour cet aspect, notamment avec l'utilisation d'auto-encodeurs dans (Li et al., 2020) (l'utilisation de ces méthodes reste pour le moment limitée). Enfin, il peut être également utile de comparer des cellules issues d'expériences différentes, provenant par exemple de différents laboratoires, ou encore capturées à partir de différents protocoles expérimentaux. Des méthodes d'intégration de données (*data integration*) sont alors utilisées afin de traiter ces différentes cellules comme un unique jeu de données.

Exploitation des données

Même si les méthodes d'analyse employées pour l'exploitation des données *single-cell* dépendent du type d'expérimentation et de ses objectifs, on retrouve généralement plusieurs étapes communes telles que la sélection de variables, la réduction de dimensionnalité, le clustering des cellules et les tests d'expression différenciée. Les outils complets d'analyse de données *single-cell* tels que *Seurat* implémentent généralement chacune des ces étapes de calcul. La **sélection de variables** permet dans un premier temps de réduire la dimensionnalité du jeu de données afin d'en réduire le coût d'analyse. Celle-ci est généralement effectuée en sélectionnant les gènes les plus variables dans les données. En effet, on peut supposer que les gènes dont l'expression est presque constante à travers les cellules n'apportent pas beaucoup d'information sur les caractéristiques biologiques des données. La **réduction de dimensionnalité** consiste ensuite à transformer les données en un ensemble de plus petite dimension afin d'en faciliter la visualisation et de mieux en comprendre la variabilité. Généralement, cette réduction de dimensionnalité est réalisée à l'aide de méthodes linéaires telles que l'Analyse en Composantes Principales (ACP) combinées à des méthodes non linéaires comme la méthode *t-SNE* (Van der Maaten et al., 2008) ou encore la méthode *UMAP* (McInnes et al., 2018). L'utilisation de méthodes non linéaires retire cependant l'interprétabilité des données. Le **clustering des cellules** a pour but de détecter plusieurs populations de cellules différentes à l'intérieur d'un jeu de données. En effet, les cellules séquencées sont généralement des cellules de plusieurs types

différents, avec des profils d'expression génétique variés. Afin de réaliser le clustering des cellules, un **graphe de voisinage** est dans un premier temps calculé à partir des cellules. Ce graphe [4] connecte les cellules les plus similaires en terme d'expression des gènes. Le clustering des cellules est alors réalisé en détectant des ensembles de cellules particulièrement connectées à l'intérieur du graphe, ce qui est réalisé à l'aide d'algorithmes tels que *Louvain* (Blondel et al., 2008). Les méthodes d'analyse de réseaux (de gènes) présentées dans la suite de ce chapitre ont été en partie inspirées de ces approches basées sur des graphes de cellules. Enfin, le **test d'expression différentielle** (*differential expression testing*) vise à déterminer, à l'aide d'un test statistique, les gènes différemment exprimés entre deux groupes de cellules. Par exemple, ces méthodes permettent de repérer les gènes ne se comportant pas de la même manière entre un groupe de données "test" et un groupe de données "contrôle". La réalisation d'un tel test produit généralement une liste de gènes, pour lesquels sont précisés le groupe de données dans lequel le gène est le plus exprimé ainsi que le pourcentage de cellules exprimant le gène pour chacun des deux groupes.

Interprétation dynamique via l'inférence de trajectoires pseudo-temporelles

Comme cela a été mentionné précédemment, la densité et la précision des données *single-cell* encourage une interprétation dynamique, en supposant que les cellules sont observées dans différents états d'un processus dynamique global (par exemple un processus de différenciation cellulaire). Les méthodes dites d'inférence pseudo-temporelles, ou encore inférence de trajectoire, proposent ainsi de "réordonner" les cellules selon un axe temporel fictif. Bien que ces méthodes soient principalement développées pour faciliter l'interprétation des données, elles présentent aussi un certain intérêt pour la modélisation qualitative en permettant de mettre les données sous forme de séries (pseudo-)temporelles. Un grand nombre de méthodes ont été évaluées dans (Saelens et al., 2019). Ces différentes méthodes reposent parfois sur quelques informations a priori comme un ensemble des cellules de "départ" pour la trajectoire à inférer, un ensemble de cellules d'"arrivée", ou encore les clusters de cellules correspondant aux différents types cellulaires. Les méthodes se distinguent par le type de topologie des trajectoires inférées : les plus simples proposent par exemple des trajectoires linéaires, c'est-à-dire un ordre total entre les cellules, et d'autres méthodes plus sophistiquées proposent des trajectoires sous la forme d'arbres, permettant de représenter des comportements plus complexes tels que des bifurcations. On notera par

4. Ce graphe permet en quelque sorte d'approximer la distribution de l'expression des gènes dans les cellules.

ailleurs que ces méthodes sont basées sur des hypothèses qui sont plus difficiles à vérifier en pratique. Les trajectoires résultantes varient donc significativement d'une méthode à l'autre, et il semblerait qu'une connaissance initiale de la topologie espérée des données soit nécessaire pour assurer un choix pertinent de la méthode à utiliser.

Une autre approche nommée *RNA-velocity* a aussi été proposée pour permettre une interprétation dynamique des données. Celle-ci consiste à exploiter l'épissage des molécules d'ARN à partir des séquences disponibles dans les données. Le rapport entre la quantité d'ARN pré-messager et celle d'ARN mature pour chaque gène permet alors d'inférer une information dynamique sur l'évolution de la quantité de molécules d'ARN pour un gène particulier dans une cellule donnée (Bergen et al., 2020 ; La Manno et al., 2018). Bien que ces méthodes comportent des défauts (Bergen et al., 2021), elles peuvent se révéler intéressantes pour la modélisation qualitative car les données dynamiques obtenues sont propres à chaque cellule, et non reconstruites comme c'est le cas des méthodes pseudo-temporelles.

Inférence de réseaux et de modèles qualitatifs

La quantité et la précision de données *single-cell* a fortement encouragé le développement de méthodes pour la modélisation des réseaux génétiques, notamment à partir de l'inférence de modèles qualitatifs. Les méthodes les plus répandues sont celles permettant l'inférence de réseaux de régulation sous la forme de graphes. De nombreuses méthodes ont été confrontées via plusieurs protocoles d'évaluation dans (S. Chen et al., 2018), (Pratapa et al., 2019) ou encore (Kang et al., 2021). Certaines méthodes d'inférence reposent sur l'utilisation des approches pseudo-temporelles mentionnées précédemment. C'est par exemple le cas de *LEAP* (Specht et al., 2016), qui vise à inférer un réseau de co-expression (c'est-à-dires des corrélations entre gènes, notion plus faible que la régulation génétique). Les méthodes *SCODE* (Matsumoto et al., 2017) et *GRISLI* (Aubin-Frankowski et al., 2020) se basent également des approches pseudo-temporelles, mais proposent l'inférence de modèles plus complexes à partir de systèmes d'équations différentielles.

Les méthodes d'inférence de modèles qualitatifs, en particulier de réseaux booléens, ont également été proposées pour modéliser la régulation génétique à partir des données de séquençage *single-cell*. Les méthodes *SCNS* (Fisher et al., 2015 ; Woodhouse et al.,

2018), *BTR* (Lim et al., 2016), *SgpNet* (Gao et al., 2020), ainsi que celle proposée dans (Hamey et al., 2017) visent à inférer des réseaux booléens asynchrones. Dans le cas de *SCNS* et *SgpNet*, les données en entrée sont sous la forme de paires de transitions discrétisées, obtenues à partir de l'inférence d'un graphe entre les cellules du jeu de données (notion similaire au graphe de voisinage). La méthode proposée dans (Hamey et al., 2017) repose sur l'utilisation d'une approche pseudo-temporelle pour l'obtention de séries temporelles. L'inférence dans *SCNS* et dans (Hamey et al., 2017) est effectuée à partir d'une recherche de satisfaction de contraintes tandis que *SgpNet* repose sur un algorithme génétique. Les données utilisées par *BTR* ne sont pas sous la forme de séries temporelles. Au contraire, un réseau booléen est inféré à l'aide d'une méthode d'optimisation heuristique et les états simulés à partir de ce réseau sont comparés aux cellules du jeu de données utilisé. La méthode *SingCellNet* (H. Chen et al., 2015) vise à inférer un réseau booléen probabiliste à l'aide d'un algorithme génétique, à partir de (courtes) séries temporelles reconstruites en utilisant des connaissances a priori sur les données. Enfin, l'algorithme proposé dans (Schwab et al., 2021) reconstitue également des courtes séries temporelles à partir de données de différents individus et infère des réseaux booléens synchrones en utilisant l'algorithme *BestFit* (Lähdesmäki et al., 2003). Par ailleurs, il faut noter que ces méthodes s'appliquent à des réseaux de petite taille (10 à 20 variables), voire de taille modérée (jusqu'à 200 variables dans le cas de la méthode proposée dans (Schwab et al., 2021)). De plus, à l'instar des méthodes d'inférence de trajectoire, les méthodes d'inférence de réseaux font appel à des hypothèses de modélisation complexes et les résultats peuvent également varier significativement en fonction des méthodes utilisées. Pour ces raisons, l'inférence de réseau n'est pas systématiquement utilisée pour l'exploitation des données *single-cell* (Luecken et al., 2019).

4.1.3 Données *single-cell* utilisées

Pour évaluer notre méthode *LOLH* sur des données réelles, nous nous intéressons à un jeu de données en particulier issu d'une collaboration avec des chercheurs de l'équipe de Mickaël Ménager à l'institut Imagine [5]. Bien que *LOLH* puisse être facilement utilisé sur d'autres jeux de données, voire d'autres types de données, l'application sur un jeu de données précis nous permet de tirer avantage de l'expertise des chercheurs impliqués dans cette collaboration. L'institut Imagine a ainsi mis à notre disposition une matrice de

5. https://www.institutimagine.org/fr

données *single-cell* contenant des cellules mononucléées sanguines périphériques (*pbmc*) extraites à partir de deux individus. Ce jeu de données contient ainsi des lymphocytes et des monocytes, qui sont des cellules impliquées dans le système immunitaire.

Le jeu de données contient 9198 cellules : 5396 cellules provenant du premier individu et 3802 cellules venant du second. Les données renseignent jusqu'à 14206 gènes, et chaque cellule contient environ 1000 gènes détectés. Ce jeu de données a été antérieurement pré-traité par l'institut Imagine à l'aide de l'outil *Seurat*. L'annexe C apporte des précisions supplémentaires sur le protocole expérimental, ainsi que sur les outils utilisés pour le prétraitement des données. Le prétraitement a ainsi permis d'identifier les différents sous-types des cellules présentes dans l'échantillon. Par exemple, on y distingue plusieurs types de lymphocytes T : les lymphocytes T cytotoxiques (*CD8*), les T auxiliaires (*CD4*) et les T régulateurs (*Treg*). La précision du séquençage *single-cell* permet également de distinguer les cellules naïves des cellules différenciées, comme c'est le cas pour les lymphocytes B dans ces données. Ces sous-types cellulaires ont été fournis avec les données d'expression génétique, sous la forme d'étiquettes pour chaque cellule renseignant le nom du type. Le prétraitement a également permis de calculer une représentation bi-dimensionnelle des données à des fins de visualisation, à partir de l'algorithme *UMAP* mentionné précédemment. Cette représentation permet de donner une intuition visuelle sur le contenu des données, notamment concernant les différents types de cellules. Celle-ci a également été fournie avec les données, sous la forme de coordonnées 2d pour chaque cellule.

La figure 4.2 permet de visualiser les cellules à partir de la représentation calculée par la méthode *UMAP*, avec des couleurs indiquant les types cellulaires obtenus via le prétraitement. Dans le cadre de notre application, nous partons d'une matrice d'expression génétique normalisée à partir de la méthode *SCTransform* (Hafemeister et al., 2019). Pour les besoins de *LOLH*, il nous a par ailleurs été nécessaire de discrétiser les données. Nous avons effectué cette discrétisation de façon empirique, à partir des distributions de valeurs des gènes (nous donnons plus de détails concernant la procédure employée dans l'annexe C). À partir de cette discrétisation, nous pouvons former les différents atomes logiques utilisés pour l'application de l'inférence logique sur ces données. De manière à faciliter l'interprétation de ces atomes, nous les noterons $X_{i/j}$ par la suite, où X désigne le gène concerné, i désigne la valeur discrète de ce gène pour cet atome et j désigne la

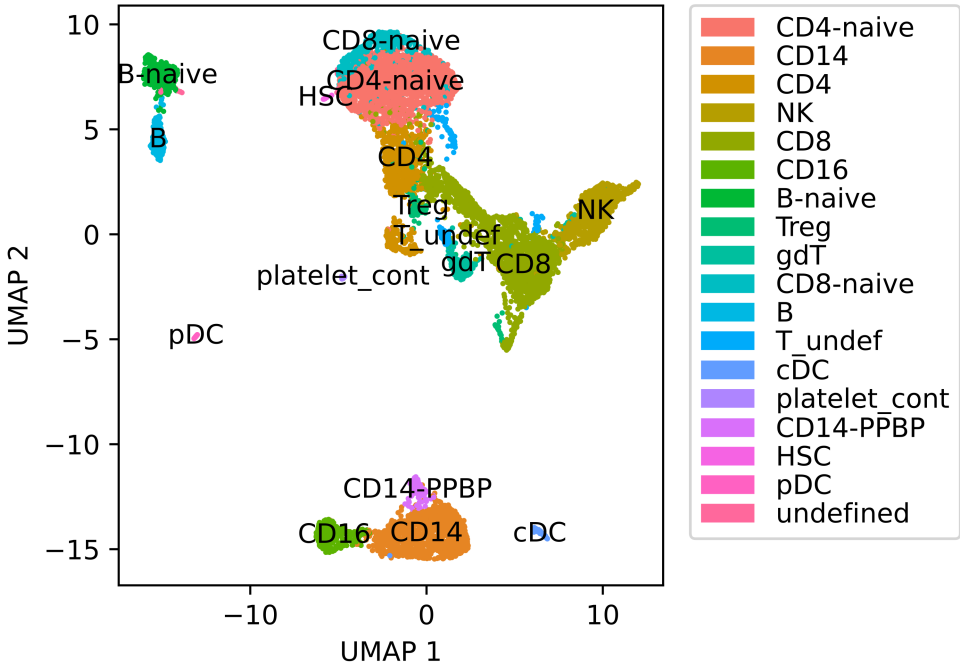

FIGURE 4.2 – Représentation 2d du jeu de données obtenue avec la méthode *UMAP*. Chaque point représente une cellule, et est coloré en fonction du type cellulaire identifié.

valeur discrète maximale que peut détenir le gène (la valeur minimale étant toujours 0). Cette notation permet ainsi de plus facilement évaluer le niveau d'activité du gène.

Nous avons également mis **en ligne** le jeu de données à l'adresse suivante : http://doi.org/10.5281/zenodo.4730807. Ce dépôt contient la matrice d'expression génétique brute, la matrice normalisée et la matrice discrétisée, permettant ainsi de reproduire les analyses effectuées dans la suite de ce chapitre. Les données sur les types cellulaires ainsi que la représentation 2d obtenue via le prétraitement sont également présentes dans ce dépôt.

4.2 Inférence de la co-expression avec *LOLH*

Puisque l'interprétation dynamique des données *single-cell* représente en soi une difficulté importante, nous identifions dans un premier temps des problèmes d'inférence "statique" afin d'évaluer *LOLH* dans un contexte concret. En effet, nous avons mis en évidence dans le chapitre 3 le fait que le framework *LFIT* consiste à modéliser le problème d'inférence de la dynamique à partir d'un ensemble de problèmes de classification binaire. En se basant sur cela, et puisque *LOLH* a été fondamentalement conçu dans le but de faire de la classification, nous identifions dans cette partie plusieurs problèmes de classification à partir des données *single-cell* fournies. Une première application consiste à considérer le problème de classification du type cellulaire à partir des informations fournies dans les données sur les types des cellules. La deuxième application vise à inférer des corrélations entre l'expression des gènes, dans le but de proposer un réseau de co-expression sur les données. Nous montrons dans ces deux applications que *LOLH* permet d'analyser de manière pertinente un jeu de données réel, et nous proposons plusieurs approches d'interprétation des résultats obtenus à partir des règles logiques inférées. Les résultats présentés dans cette partie ont été publiés dans (Buchet et al., 2021), et sont reproductibles avec notre implémentation en ligne [6].

4.2.1 Classification logique des cellules *NK*

Dans cette première application, nous formulons un problème de classification consistant à prédire le type d'une cellule à partir de son profil d'expression génétique. L'intérêt de formuler un tel problème est d'exploiter le modèle inféré par *LOLH* pour interpréter les données de façon logique. Cette interprétation nous permet à la fois d'évaluer les performances de *LOLH* sur un problème concret, et d'analyser différemment les résultats obtenus via le traitement antérieur des données (par exemple dans le but de comprendre le résultat des algorithmes de clustering ou encore la réduction de dimensionnalité des données par la méthode *UMAP*). La classification cellulaire est un exemple pertinent puisque le résultat de l'inférence peut être confronté à des connaissances biologiques telles que celles des marqueurs génétiques connus. Puisque les problèmes de classification traités par *LOLH* sont binaires, on se concentre sur la classification d'un type de cellule en particulier. Pour l'illustrer, nous sélectionnons les cellules *NK* (*natural killer*) identifiées lors du prétraitement des données (voir figure 4.2). Ce type cellulaire est intéressant pour

6. Notre implémentation est disponible à l'adresse https://github.com/smbct/LOLH.git

évaluer la méthode car l'expression génétique de ces cellules est similaire à celle des cellules *CD8*.

La formulation du problème de classification se fait à partir des étiquettes des types cellulaires fournis avec le jeu de données *single-cell*. Un problème de classification a ainsi été formulé sur les cellules discrétisées, avec 829 exemples positifs $\mathcal{S}^+_{NK}$ (cellules labélisées *NK*), et 8369 exemples négatifs $\mathcal{S}^-_{NK}$ (cellules non labellisés *NK*) sur 14206 gènes. Ainsi dans cet exemple $|\mathcal{S}^+_{NK}| = 829$ et $|\mathcal{S}^-_{NK}| = 8369$. *LOLH* appliqué sur cette instance avec un seuil de 0.55 (voir algorithme 4, page 103) a retourné une règle optimisée $\mathcal{R}_{LOLH}$ contenant 21 atomes dans son corps. Le corps de la règle obtenue contient les atomes suivants (en utilisant la notation $X_{i/j}$ introduite dans la partie 4.1.3) :
$corps(\mathcal{R}_{LOLH}) = \{$ $GNLY_{1/1}$, $GZMB_{1/1}$, $NKG7_{1/1}$, $PRF1_{1/1}$, $KLRD1_{1/1}$, $CTSW_{1/1}$, $KLRF1_{1/1}$, $CST7_{1/1}$, $GZMA_{1/1}$, $HOPX_{1/1}$, $KLRB1_{1/1}$, $IL2RB_{1/1}$, $TYROBP_{1/1}$, $CD7_{1/1}$, $CD247_{1/1}$, $CMC1_{1/1}$, $CLIC3_{1/1}$, $SPON2_{1/1}$, $MATK_{1/1}$, $FCER1G_{1/1}$, $B2M_{2/3}$ $\}$
On remarque que tous les atomes sélectionnés pour l'inférence de la règle ont une valeur discrète non nulle, ce qui correspond dans les données à des gènes exprimés.

Analyse de la règle sur les données

Pour analyser la règle logique inférée, on peut se concentrer sur son erreur de couverture et sur les scores de ses atomes par rapport aux données de l'instance de classification. La figure 4.3 permet de visualiser les erreurs positives et négatives des atomes pour la classification des cellules *NK*. On remarque le même profil que sur les données artificielles traitées avec *LOLH* dans le chapitre 3. On remarque également que la majorité des atomes ont un score insuffisant (proche de 0) pour classifier efficacement les cellules. Cependant, les atomes sélectionnés par *LOLH* ont des scores élevés, qui permettent à la règle d'obtenir un bon score de classification.

Une deuxième manière d'explorer la règle consiste à étudier les histogrammes de son erreur de couverture sur les données, comme cela a été proposé dans le chapitre 3. Sur la figure 4.4, on peut observer une assez bonne séparation des histogrammes entre les exemples négatifs et les examples positifs. Il y a cependant un chevauchement de l'erreur de couverture entre certains exemples positifs et certains négatifs, pour des erreurs comprises entre 5 et 14. Cela laisse penser que la règle $\mathcal{R}_{LOLH}$ est trop générale pour classifier parfaitement toutes les cellules, et que certaines restent donc difficiles à caractériser.

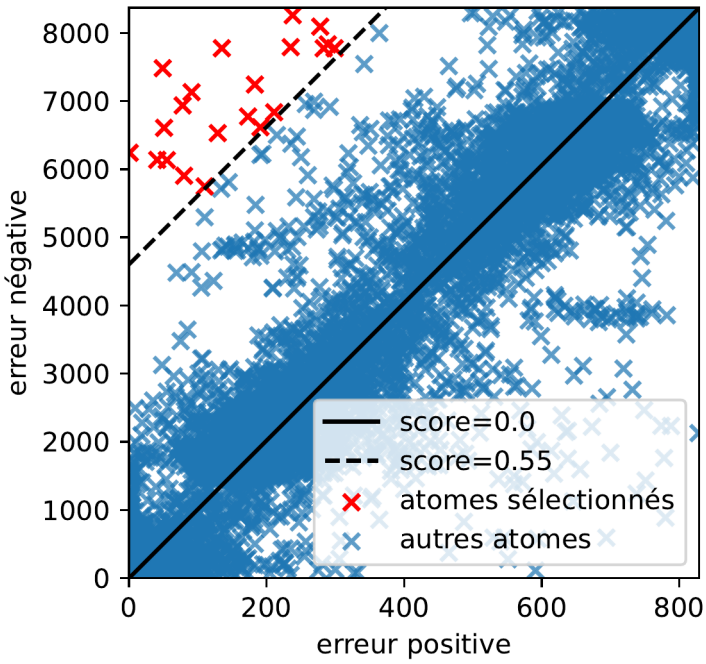

FIGURE 4.3 – Erreurs positive et négative des atomes pour l'instance de classification du type cellulaire *NK*. Les atomes sélectionnés dans la règle de *LOLH* sont en rouge.

De manière intéressante, l'histogramme des exemples négatifs laisse observer deux populations de cellules différentes selon l'erreur de couverture. On peut tenter de visualiser cela plus en détail sur la figure 4.5 en utilisant la représentation 2d des données et en colorant les cellules en fonction de leur erreur de couverture. Si on compare cette dernière à la figure 4.2, on remarque que les cellules pour lesquelles l'erreur est minimale correspondent globalement aux cellules étiquetées *NK*, ce qui est cohérent avec les histogrammes de l'erreur de couverture en figure 4.4. Par ailleurs, on observe que certaines des cellules non étiquetées *NK* présentent également une faible erreur de couverture. C'est par exemple le cas de quelques cellules étiquetées myéloïdes (en bas de la représentation sur la figure C.1, en annexe C) ou encore de quelques lymphocytes *B* (en haut à gauche de la figure C.1). On remarque également que la majorité des cellules *CD8* ont une erreur assez faible. Cela semble confirmer la ressemblance entre les cellules *CD8* et *NK* en terme d'expression génétique.

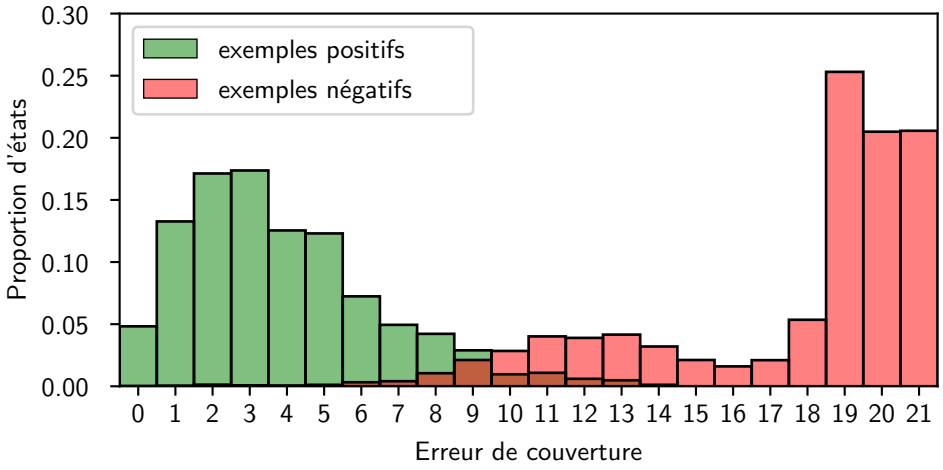

FIGURE 4.4 – Histogrammes de l'erreur de couverture de la règle de *LOLH* sur les exemples positifs (en vert) et sur les exemples négatifs (en rouge).

Vérification par analyse d'expression différentielle et par marqueurs

Pour confronter plus en détail le résultat de *LOLH* avec l'analyse effectuée sur les données, il est possible de se baser sur des outils d'analyse de données *single-cell* existants, comme l'analyse différentielle d'expression mentionnée dans la sous-partie 4.1.2. Étant donnés deux groupes de cellule, les outils d'analyse différentielle calculent les gènes qui sont le plus différemment exprimés entre ces deux groupes. L'analyse d'expression différentielle a été appliquée sur les données avec *Seurat* (avec la function *FindMarkers*), en utilisant les cellules étiquetées *NK* en tant que premier groupe, et les autres cellules comme second groupe (le script utilisé pour l'analyse est disponible dans l'implémentation *LOLH*). Le tableau résultant, contenant les 30 gènes les plus différemment exprimés triés selon la troisième colonne, peut être trouvé en Annexe E.1. Dans ce tableau, la deuxième et la dernière colonne indiquent la *significativité* statistique du résultat. Les quatrième et cinquième colonnes indiquent le pourcentage de cellules dans lesquelles le gène considéré est trouvé respectivement pour le premier et le deuxième groupe de cellules. Enfin, la troisième colonne indique à quel point le gène est différemment exprimé entre les groupes de cellules (des valeurs positives indiquent que le gène est plus exprimé dans le premier groupe, c'est-à-dire dans le groupe de cellules étiquetées *NK* dans ce contexte) [7]. On peut remarquer que la plupart des gènes présents dans $\mathcal{R}_{LOLH}$ sont dans le tableau, à l'excep-

7. Plus de détails sur l'analyse d'expression différentielle sont disponibles dans la documentation de *Seurat*, à l'adresse https://satijalab.org/seurat/reference/findmarkers.

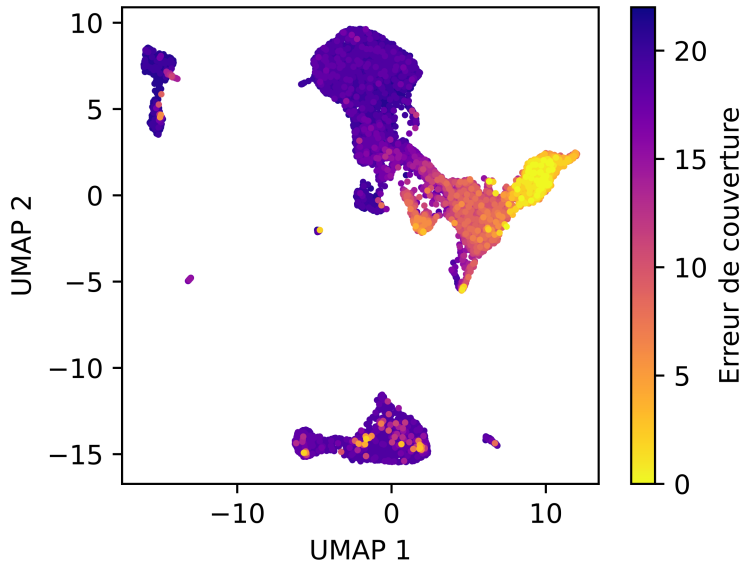

FIGURE 4.5 – Visualisation de l'erreur de couverture de la règle de *LOLH* sur la représentation 2d des cellules.

tion de *TYROBP*, *MATK*, *FCER1G* et *B2M*. Cela laisse à penser que les atomes ayant des scores élevés correspondent effectivement aux gènes différemment exprimés entre les exemples positifs et les exemples négatifs.

Une autre manière d'évaluer la pertinence de la règle inférée par *LOLH* consiste à comparer les gènes sélectionnés à des marqueurs biologiques connus, ici dans le cas des cellules *NK*. Cette confrontation permet à la fois de confirmer l'efficacité de *LOLH*, et de confirmer que l'étiquetage des cellules *NK* du jeu de données effectué antérieurement a été correctement réalisé. L'annexe D contient une liste de gènes associés aux différents types de cellules. Cette liste a été proposée par les chercheurs de l'institut Imagine impliqués dans ce travail. Les gènes *NKG7*, *PRF1* et *IL2RB* sont effectivement principalement présents dans les cellules *NK*. De plus, d'autres gènes, qui sont généralement exprimés dans les cellules *NK* (mais pas seulement) sont présents, tels que *GNLY*, *GZMB*, *KLRD1*, *CST7* et *FCER1G*.

4.2.2 Inférence de réseaux de co-expression

Dans un second temps, on peut utiliser la valeur discrète d'un gène pour formuler un nouveau problème de classification où les exemples positifs correspondent aux cellules dans lesquelles cette valeur est vérifiée, et où les exemples négatifs correspondent aux autres cellules. Ce type de problème de classification peut être formulé pour chaque valeur discrète de l'ensemble des gènes présents dans les données (ou encore pour chaque atome logique). Cette formulation permet ainsi de détecter de manière non supervisée des corrélations entre les différents gènes du jeu de donnée afin de mettre en évidence d'éventuelles propriétés biologiques. Ce type d'analyse, que l'on appelle aussi analyse de la co-expression, est généralement réalisé à partir de la construction de graphes, et permet par exemple de mettre en évidence des relations de régulation ou encore de signalisation entre différents gènes (Fuller et al., 2011). Nous combinons ainsi ici l'utilisation de *LOLH* avec des méthodes d'analyse de graphes, dans une approche inspirée de celle des graphes de voisinage (de cellules) utilisés pour le clustering des données *single-cell*. En comparaison aux outils d'analyse de données *single-cell* mentionnés dans la sous-partie 4.1.2 en page 124, cette analyse se révèle intéressante car les calculs sont effectués au niveau des gènes, et non au niveau des cellules.

Construction d'un graphe de co-expression à partir de *LOLH*

Étant donné un atome x_i représentant la valeur discrète i du gène x dans le jeu de données, on peut former un problème de classification des cellules pour *LOLH* de la manière suivante : $\mathcal{S}^+_{x_i} = \{s \in \mathcal{S} \mid s(x) = x_i\}$ and $\mathcal{S}^-_{x_i} = \mathcal{S} \setminus \mathcal{S}^+_{x_i}$, où $\mathcal{S}$ désigne l'ensemble des cellules discrétisées du jeu de données. Ce problème ainsi formulé permet de détecter, à partir de l'inférence de règles logiques, les gènes dont l'expression est corrélée à l'atome x_i. Pour ce problème de classification ainsi formulé à partir de l'atome x_i, le gène x est retiré des données pour que celui-ci ne soit pas utilisé dans l'inférence de règle (l'atome x_i étant corrélé à lui-même). L'application de *LOLH* sur $\mathcal{S}^+_{x_i}$ et $\mathcal{S}^-_{x_i}$ résulte ainsi en une règle logique $\mathcal{R}_{x_i}$ obtenue pour l'atome x_i à partir d'un seuil choisi, et où $corps(x_i)$ contient les atomes les plus corrélés à x_i. Dans cette application, il est également intéressant d'extraire le score de chaque atome sélectionné par *LOLH* afin de distinguer les différents degrés de corrélation des atomes.

En répétant ce schéma de modélisation pour chaque atome logique des données, on peut alors interpréter les règles résultantes comme les arêtes d'un graphe non orienté sur l'ensemble des atomes. Un tel graphe possède ainsi une arête entre un atome x_i et un atome y_j si les deux atomes sont corrélés, c'est-à-dire si $x_i \in \mathcal{R}_{y_j}$ et si $y_j \in \mathcal{R}_{x_i}$. Cette arête, si elle existe, peut alors être pondérée en moyennant les scores des deux atomes dans leurs instances respectives. On obtient ainsi un graphe pouvant contenir jusqu'à la totalité des atomes logiques, et dont les arêtes témoignent des différentes corrélations entre les atomes. Une sélection est de plus réalisée sur ce graphe afin de ne pas prendre en compte les corrélations qui ne sont pas assez fortes (sélection sur les scores des atomes via $LOLH$) et également afin de ne pas prendre en compte les gènes qui sont presque constants dans les données (sélection à partir de la proportion entre $\mathcal{S}^+$ et $\mathcal{S}^-$).

Identification de propriétés biologiques par clustering

La méthodologie introduite précédemment nous a ainsi permis de générer un graphe de corrélation (ou co-expression) pondéré sur les atomes à partir du jeu de données *single-cell* discrétisé. Pour la construction de ce graphe, un seuil de 0.35 a été utilisé sur le score des atomes sélectionnés dans $LOLH$, et les gènes variants dans moins de 200 cellules ont été ignorés. Les gènes mitochondriaux et ribosomaux ont également été ignorés car ceux-ci sont généralement exprimés dans la majorité des cellules, ne sont donc pas informatifs pour l'exploitation des données.

Il est possible d'analyser le graphe obtenu en appliquant une méthode de clustering de graphe sur les **atomes**. Nous avons utilisé l'algorithme de clustering de graphe *Louvain* (Blondel et al., 2008) dans ce but, à partir de l'implémentation python *python-louvain*[8]. Dans ce graphe, l'application de la méthode *Louvain* nous a permis d'obtenir 6 clusters d'atomes. Le contenu de chacun de ces clusters peut être trouvé en annexe F. Des symétries sont observées entre les clusters, c'est-à-dire que certains clusters contiennent globalement des atomes correspondant aux mêmes gènes, avec des valeurs d'expression discrètes opposées. Ainsi, les paires de clusters suivantes peuvent être associées ensemble : clusters 0 et 1, clusters 2 et 3, et clusters 4 et 5. On peut remarquer que les clusters 0, 3 et 4 contiennent majoritairement des atomes avec une valeur discrète nulle. Or comme nous l'avons mentionné précédemment, les gènes sont, en *single-cell*, détectés dans peu de cellules et les atomes présentant une valeur discrète nulles sont donc peu informatifs sur

8. *python-louvain* est disponible à l'adresse https://github.com/taynaud/python-louvain.git

l'activité des gènes. Pour cette raison, nous nous concentrons par la suite sur l'analyse des clusters 1, 2 et 5, dont les atomes correspondent majoritairement à des gènes exprimés.

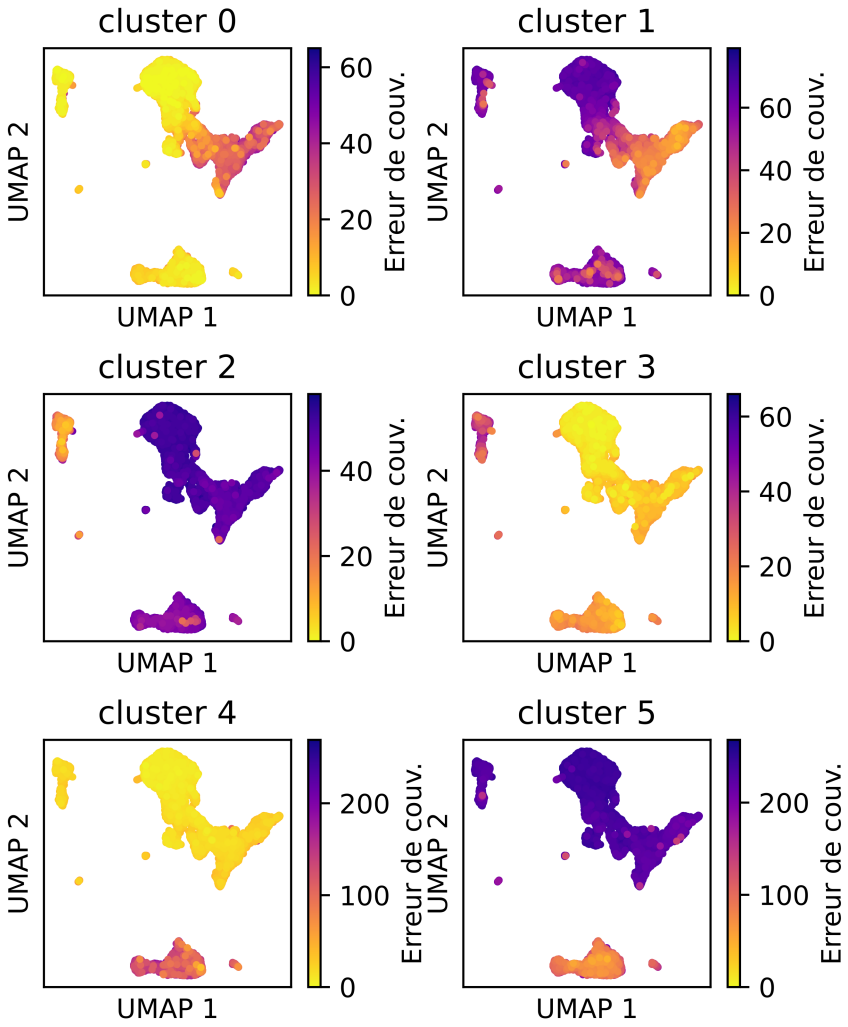

FIGURE 4.6 – Visualisation sur la représentation 2d de l'erreur de couverture de chaque cluster sur l'ensemble des cellules à partir de couleurs.

Nous pouvons ensuite tenter d'identifier les cellules que caractérisent ces différents clusters d'atomes, c'est-à-dire les cellules pour lesquelles les valeurs discrètes des atomes sont majoritairement vérifiées. Pour cela, nous interprétons les clusters d'atomes comme le corps de règles logiques. Il est alors possible de calculer l'erreur de couverture des différents clusters sur chaque cellule (à partir de la définition 3.5 en page 98). En procédant ainsi, on peut alors considérer que les cellules ayant la plus petite erreur de

couverture pour un cluster d'atomes donné sont représentatives de ce cluster. La figure 4.6 permet de visualiser les erreurs de couverture des clusters sur l'ensemble des cellules à partir de couleurs sur la représentation 2d du jeu de données (voir partie 4.1.3). Dans cette figure, les symétries sont clairement visibles. On peut également associer les clusters d'atomes à certains types cellulaires identifiés dans les figures 4.2 en page 131 et C.1 en page 172, en repérant les atomes dont les gènes sont des marqueurs biologiques. Le tableau en annexe D permet d'effectuer cette identification. Ainsi, le cluster 1 semble correspondre aux lymphocytes T différenciés et aux cellules NK car il contient les atomes suivants : $GNLY_{1/1}$, $IL2RB_{1/1}$, $GZMB_{1/1}$, $CST7_{1/1}$, $NKG7_{1/1}$, $CCL5_{1/1}$, $KLRD1_{1/1}$, $PRF1_{1/1}$, $GZMH_{1/1}$, $CD8A_{1/1}$ et $CD8B_{1/1}$. Le cluster 5 peut également être associé aux *myéloïdes*, ce que l'on peut confirmer par la présence de $AIF1_{1/1}$, $CD14_{1/1}$, $CST3_{1/1}$, $FCER1G_{1/1}$, $FCN1_{1/1}$, $LGALS3_{1/1}$, $LST1_{1/1}$, $LYZ_{1/1}$, $S100A4_{2/2}$, $S100A9_{1/1}$ et $TYROBP_{1/1}$, ainsi que par les atomes suivants indiquant l'absence de gènes associés aux lymphocytes T : $CD3E_{0/2}$, $IL32_{0/2}$, et $IL7R_{0/1}$. Enfin, le cluster 2 correspond aux lymphocytes B, ce qui est confirmé par la présence de $BLK_{1/1}$, $CD37_{2/2}$, $CD74_{1/1}$, $CD79A_{1/1}$, $CD79B_{1/1}$, $HLA\text{-}DMA_{1/1}$, $HLA\text{-}DQA1_{1/1}$, $MS4A1_{1/1}$, $PAX5_{1/1}$, $SPIB_{1/1}$, $TCL1A_{1/1}$ et $VPREB3_{1/1}$.

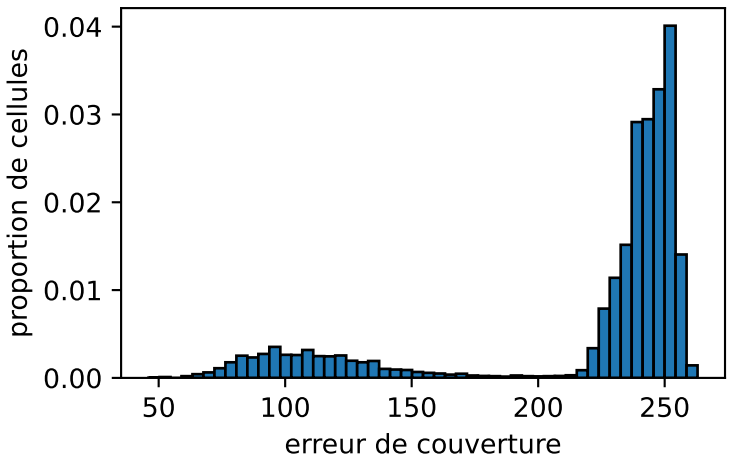

FIGURE 4.7 – Histogramme de l'erreur de couverture du cluster 5 sur l'ensemble des cellules.

Toutefois, la figure 4.6 laisse penser que les clusters 1 et 2 caractérisent également des cellules qui ne sont pas étiquetées respectivement T et NK ou B. En effet, certaines cellules situées en bas de la représentation 2d ont une erreur de couverture faible pour ces deux

clusters. Or, ces cellules sont celles associées aux myéloïdes (voir figure C.1 en annexe C), qui ne devraient donc pas présenter de marqueurs génétiques pour d'autres types de cellules. Pour étudier cela plus en détail, il est possible de calculer un nouveau graphe de co-expression plus spécifique en effectuant une sélection de cellules. Nous illustrons donc ceci à partir du cluster 5 caractérisant les myéloïdes, afin de vérifier si l'ensemble des cellules étiquetées myéloïdes ne présentent pas de marqueurs liés à d'autres types cellulaires. Nous construisons donc un sous-jeu de données en sélectionnant les cellules ayant une erreur de couverture ≤ 193 sur le cluster 5, ce qui peut être visualisé sur l'histogramme de la figure 4.7 (où l'on discerne deux populations de cellules).

Construction d'un sous-graphe à partir du cluster 5

En se basant sur la sélection de cellules proposée précédemment, nous avons créé un nouveau graphe en considérant seulement les cellules correspondant au cluster 5. Ce cluster présentant des marqueurs génétiques du type myéloïde, on s'attend donc à ce qu'un graphe de co-expression inféré à partir de ces cellules permette de détecter des clusters d'atomes relatifs aux sous-types de myéloïde (c'est-à-dire *CD14*, *CD16* et *cDC*, voir figure 4.2 en page 131). L'application de l'algorithme *Louvain* sur ce nouveau graphe de co-expression donne 8 clusters d'atomes. Puisque des symétries peuvent également être observées, nous nous concentrons sur l'analyse des clusters 2, 3, 5, et 6, qui peuvent être trouvés en annexe F.2. De manière similaire à l'analyse précédente, l'erreur de couverture entre ces clusters et les cellules peut être visualisée sur la figure 4.8 (18 cellules ont été ignorées sur la figure pour des raisons de clarté).

En utilisant les marqueurs génétiques présents en annexe D, le cluster 2 peut être associé au type cellulaire *CD16*, par la présence de $CD14_{0/1}$, $CSF1R_{1/1}$, $FCGR3A_{1/1}$, $LYZ_{0/1}$, $S100A4_{2/2}$ et $S100A9_{0/1}$. Le cluster 6 peut être associé à des cellules de type *CD8* ou encore de type *NK*, avec $CD3E_{1/2}$, $IL32_{1/2}$, $IL7R_{1/1}$, $LEF1_{1/1}$, $CST7_{1/1}$, $CCL5_{1/1}$ et $NKG7_{1/1}$. Enfin, le cluster 3 peut être associé aux cellules *cDC*, avec $CD74_{1/1}$, $FCER1A_{1/1}$, et $HLA\text{-}DQA1_{1/1}$. On remarque donc que le cluster 6 ne correspond pas aux *myéloïdes* mais à un sous-type de lymphocytes *T*. Ceci est visible lorsque l'on observe l'erreur de couverture par rapport aux différents clusters sur l'ensemble des cellules du jeu de données sur la figure 4.9 car les cellules ayant la plus petite erreur de couverture sur le cluster 6 ne correspondent pas à celles qui ont été étiquetées myéloïdes (voir figure C.1).

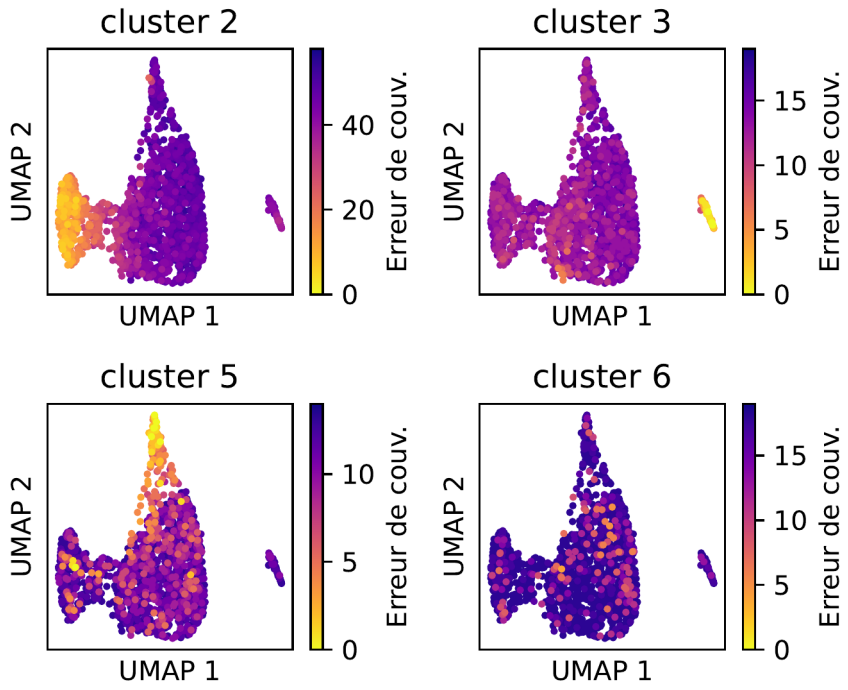

FIGURE 4.8 – Visualisation de l'erreur de couverture des atomes de chaque cluster sur l'ensemble des cellules sélectionnées.

Puisque le nouveau graphe de co-expression a été calculé sur une sous-sélection de cellules, cela indique que la sélection que nous avons effectuée n'a pas permis d'exclure rigoureusement toutes les cellules qui ne sont pas des myéloïdes. Cependant, la figure 4.9 montre que ces cellules associées au type *CD8* font également partie des cellules classifiées *CD14* et *CD16* par l'institut Imagine. Cela semble donc également montrer les limites des méthodes couramment employées pour déterminer les types cellulaires à partir de données *single-cell*, ce qui peut être causé ici par des ressemblances entre les cellules qui sont dues à d'autres marqueurs génétiques. Notre approche, qui présente tout de même aussi des limites associées à la stabilité des algorithmes de clustering, ainsi qu'au choix des paramètres, semble être complémentaire des outils classiques car elle a l'avantage de permettre l'association d'une même cellule à plusieurs groupes de gènes différents, à la différence des méthodes de clustering sur les cellules. Pour vérifier plus rigoureusement que certaines des cellules classifiées comme des myéloïdes sont effectivement des cellules appartenant à d'autres types, on peut se concentrer sur ces cellules et effectuer une sous-sélection de cellules à partir de l'erreur de couverture du cluster 6. Nous avons ainsi effectué une analyse d'expression différentielle de manière similaire à ce qui a été fait en sous-partie 4.2.1, entre

cette sélection de cellules et entre les autres cellules étiquetées *myéloïdes*. Le résultat de l'analyse en annexe E.2.1 semble confirmer que les cellules sélectionnées ne sont effectivement pas des myéloïdes, avec notamment la non-expression de *LYZ*, *LST1* et *TYROBP*.

Enfin, le cluster 5 présente également un fort intérêt puisqu'il caractérise certains myéloïdes, mais également certains lymphocytes *T* (étiquetées *T_undef* par l'institut Imagine, voir figure 4.2), comme on peut le voir sur la figure 4.9. Ce cluster contient le gène *PPBP*, un marqueur de facteur de croissance pour les plaquettes, mais qui peut également être trouvé dans les myéloïdes. Afin de vérifier si les cellules étiquetées *CD14* présentant ce marqueur sont effectivement des myéloïdes, il est possible encore une fois de procéder à une analyse d'expression différentielle entre les myéloïdes caractérisées par le cluster 5 et les autres cellules également caractérisées par le cluster 5, qui semblent correspondre à des lymphocytes *T*. Le résultat de l'analyse, présent en annexe E.2.2 indique que les cellules testées correspondent effectivement à des monocytes. La présence de *PPBP* dans ces cellules, ainsi que sa présence dans les autres cellules suggère alors qu'il pourrait être un signe d'inflammation (Elmesmari et al., 2016 ; Schwartzkopff et al., 2012).

4.3 Inférence de la dynamique avec *LOLH*

Comme nous l'avons indiqué dans la partie 4.1, les données *single-cell* n'apportent pas de réelle information temporelle sur les cellules étudiées. Toutefois, puisque les mécanismes génétiques à l'œuvre au sein de cellules sont par nature dynamiques, on peut alors tenter d'interpréter les cellules obtenues comme les différents états dynamiques d'un même processus afin d'appliquer des méthodes d'inférence de modèles qualitatifs.

Puisque nous nous concentrons sur l'application de *LOLH* sur ces données, notre objectif ici consiste à reconstruire un ensemble de transitions afin d'obtenir la forme des données attendue pour la modélisation avec le framework *LFIT* (voir partie 3.2.1). Plusieurs approches peuvent être considérées pour l'extraction de transitions. Comme nous l'avons vu en sous-partie 4.1.2, certaines méthodes d'inférence de modèles qualitatifs reposent sur l'utilisation des méthodes pseudo-temporelles mentionnées en sous-partie 4.1.2. Dans ce travail, on se concentre sur l'exploitation d'un graphe de voisinage afin d'en extraire des transitions. Ce type de graphe, généralement inféré pour l'analyse des données

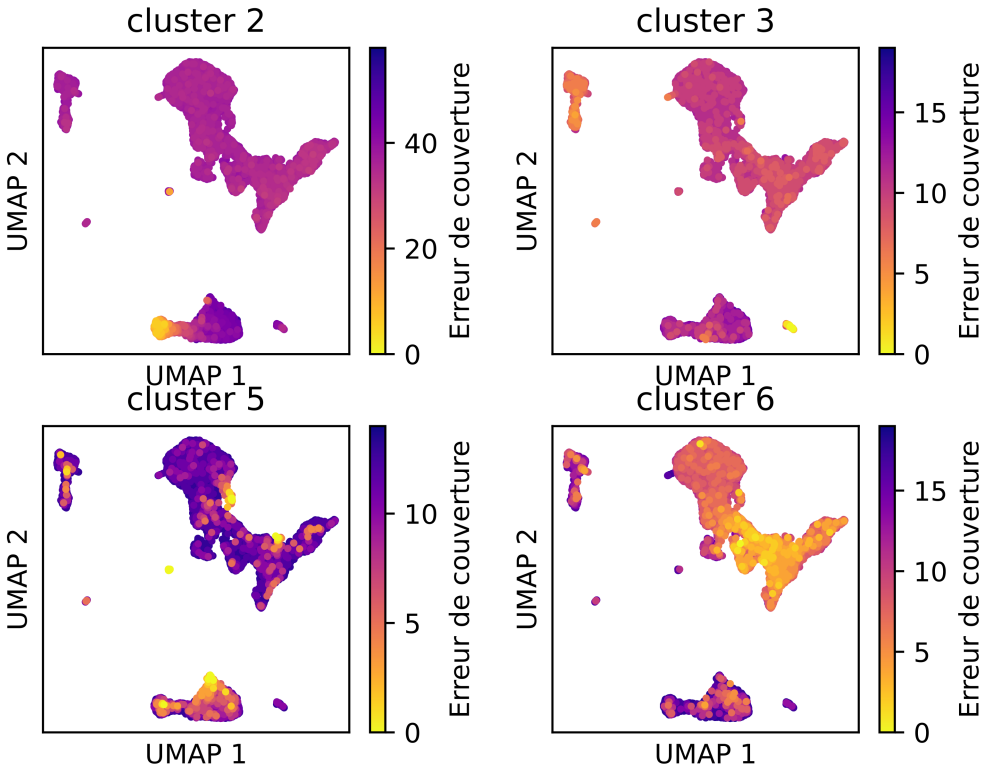

FIGURE 4.9 – Visualisation de l'erreur de couverture des clusters sur la totalité des cellules du jeu de données.

single-cell constitue un graphe (non orienté) des cellules dans lequel deux cellules sont connectées par une arête si les deux cellules sont suffisamment similaires. Bien qu'un tel graphe ait l'inconvénient de ne pas être orienté, un de ses avantages est la ressemblance des cellules connectées, ce qui facilite la recherche de corrélations entre les gènes qui est calculée par *LOLH*. L'approximation de la dynamique à partir de transitions extraites du graphe de voisinage est représentée de manière schématique en figure 4.10.

Dans la sous-partie suivante, nous proposons une stratégie pour l'extraction de transitions à partir d'un graphe de voisinage calculé sur le jeu de données. Pour cela, nous introduisons plusieurs paramètres, et nous proposons une méthode pour estimer des valeurs de paramètres intéressantes, permettant d'obtenir les transitions les plus pertinentes. Ensuite, nous appliquons *LOLH* sur les transitions obtenues et nous analysons le réseau obtenu.

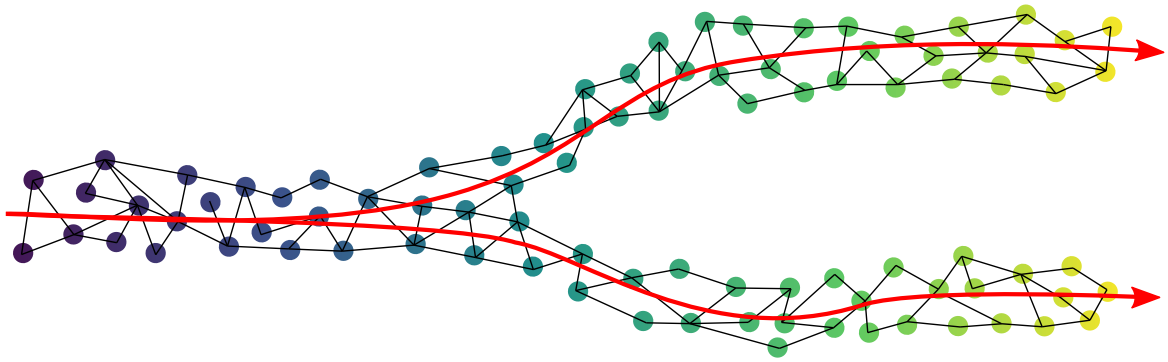

FIGURE 4.10 – Représentation schématique de l'interprétation de transitions à partir de données *single-cell*. On suppose qu'un processus dynamique global est visible, et qu'il est possible d'interpréter les cellules similaires (voisines) comme des transitions possibles. Le dégradé de couleur représente un processus global (par exemple la différenciation cellulaire) avec un point de départ (en bleu) et deux points d'arrivées possibles (en jaune). En l'absence d'information supplémentaire, les arêtes ne peuvent être orientées.

4.3.1 Extraction des transitions et formulation *LOLH*

Pour l'obtention des transitions à partir des données *single-cell*, nous nous reposons directement sur la méthode d'inférence de graphe de voisinage dans l'outil *Seurat*, via la fonction *FindNeighbors* [9]. Puisque le graphe est non orienté, une arête $\{s, s'\}$ du graphe est alors interprétée comme deux transitions (s, s') et (s', s). En effet, aucune information apparente n'est disponible pour décider si une des deux transitions est plus vraisemblable. Intuitivement, la considération des deux transitions entraîne alors l'inférence d'un modèle qui a plusieurs directions d'évolution (par exemple dans le cas d'un processus de différentiation, le modèle représenterait l'évolution des cellules dans le sens de la maturation mais aussi dans le sens contraire). Le script R d'extraction de transitions à partir du graphe est disponible avec l'implémentation en ligne de *LOLH* (dans la partie exemple sur les données de l'institut Imagine).

Formulation des problèmes de classification

Une fois que les transitions sont extraites, la difficulté suivante consiste à identifier de manière pertinente les exemples positifs $\mathcal{S}^+_{v_i}$ et les exemples négatifs $\mathcal{S}^-_{v_i}$ à partir des transitions pour l'inférence de règles logiques de chaque atome v_i avec *LOLH*. Dans le cadre des algorithmes *GULA* et *PRIDE*, un exemple s pour un atome v_i est considéré comme positif s'il existe au moins une transition (s, s') telle que $s'(v) = v_i$. Plus formellement,

9. Plus d'informations à l'adresse https://satijalab.org/seurat/reference/findneighbors.

étant donné un ensemble de transitions $\mathcal{T}_{obs}$: $\mathcal{S}^+_{v_i} = \{s \mid \exists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}$ et $\mathcal{S}^-_{v_i} = \{s \mid \nexists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}$ (voir les algorithmes 2 et 3 en page 90). Cependant, cette dernière formulation a été pensée dans le cadre des systèmes déterministes ciblés par *GULA* et *PRIDE* (Ribeiro et al., 2018b). Dans notre application de *LFIT*, nous ne pouvons faire l'hypothèse du déterminisme du système puisque les transitions sont ici extraites à partir du graphe de transition (un sommet a généralement plusieurs successeurs). Nous proposons donc une manière alternative pour former $\mathcal{S}^+$ et $\mathcal{S}^-$, basée sur l'introduction de paramètres supplémentaires. Étant donné un ensemble de transitions $\mathcal{T}_{obs}$ et un atome v_i, nous introduisons les trois paramètres τ, ρ et δ décrits par la suite, afin de déterminer si un état s prédécesseur de transitions doit être considéré comme un exemple positif ou négatif. L'algorithme 5 en page 147 formalise la procédure utilisée pour la formation de $\mathcal{S}^+_{v_i}$ et $\mathcal{S}^-_{v_i}$ à partir de ces paramètres. Nous le présentons de manière intuitive ci-dessous.

- $\boldsymbol{\tau}$: Puisqu'un état s donné peut posséder plusieurs successeurs dans les transitions $\mathcal{T}_{obs}$, il peut être utile d'ajouter une contrainte sur la proportion d'états successeurs vérifiant la présence de l'atome v_i à apprendre pour considérer s comme un exemple positif. Nous introduisons donc un paramètre $\boldsymbol{\tau}$ désignant un seuil sur la proportion des successeurs de s vérifiant v_i :
$s \in \mathcal{S}^+_{v_i} \iff (|\{s' \mid \exists (s, s') \in \mathcal{T}_{obs} : s'(v) = v_i\}|) \,/\, (|\{s' \mid \exists (s, s') \in \mathcal{T}_{obs}\}|) \geq \tau$. Nous fixons $\tau \in \{\varepsilon\} \cup [0, 1]$, où la valeur $\varepsilon \in \mathbb{R}$ désigne un nombre strictement positif arbitrairement petit, modélisant le fait qu'au moins un successeur vérifiant v_i est nécessaire pour considérer s comme positif (ce qui revient à la définition employée dans les algorithmes *GULA* et *PRIDE*).

- $\boldsymbol{\rho}$: Pour l'inférence d'un atome v_i, il peut aussi être intéressant de contraindre la valeur de la variable v dans l'état prédécesseur s d'une transition (s, s'), par exemple en forçant $s(v) = v_i$ ou encore en forçant $s(v) \neq v_i$. Nous introduisons donc un paramètre $\boldsymbol{\rho} \in \{0, 1, 2\}$ sur le prédécesseur des transitions : $s \in \mathcal{S}^+_{v_i} \implies s(v) \neq v_i$ si $\rho = 1$ et $s \in \mathcal{S}^+_{v_i} \implies s(v) = v_i$ si $\rho = 2$. La valeur 0 signifie qu'aucune contrainte n'est fixée.

- $\boldsymbol{\delta}$: Enfin, il est possible que les cellules reliées par une arête dans le graphe de voisinage soient extrêmement proches en termes d'expression génétique. Cela peut se révéler problématique pour l'extraction de transitions car il se pourrait que cette proximité complique la modélisation de l'évolution de l'expression des gènes. Nous introduisons donc

un paramètre $\boldsymbol{\delta}$ s'apparentant à un délai dans le graphe de voisinage pour obtenir des transitions plus "éloignées". Nous choisissons $\delta \in \{1, 2\}$ tel que $\mathcal{T}_{obs,\delta} = \mathcal{T}_{obs}$ si $\delta = 1$, et $\mathcal{T}_{obs,\delta} = \{(s, s'') \mid \exists (s, s'), (s', s'') \in \mathcal{T}_{obs}\}$ si $\delta = 2$.

Données : $\mathcal{T}_{obs} \subset \mathcal{S} \times \mathcal{S}$, $v_i \in \mathcal{A}_v$ l'atome à apprendre, τ, ρ, δ
Résultat : $\mathcal{S}^+_{v_i}$, $\mathcal{S}^-_{v_i}$ les exemples positifs et négatifs pour v_i

si $\delta = 2$ **alors**
  $\mathcal{T}_{obs} \leftarrow \{(s, s'') \mid \exists (s, s') \in \mathcal{T}_{obs}, \exists (s', s'') \in \mathcal{T}_{obs}\}$
fin
$\mathcal{S}^+_{v_i} \leftarrow \varnothing$
$\mathcal{S}^-_{v_i} \leftarrow \varnothing$
pour $s \in \{s_1 \mid \exists (s_1, s_2) \in \mathcal{T}_{obs}\}$ **faire** `// Pour chaque état prédécesseur`
  $positif \leftarrow \bot$
  si $\{s_2 \mid \exists (s, s_2) \in \mathcal{T}_{obs}\} \neq \varnothing$ **alors**
    si $\rho = 0 \vee (\rho = 1 \wedge s(v) \neq v_i) \vee (\rho = 2 \wedge s(v) = v_i)$ **alors**
      si $\tau = \varepsilon$ **alors**
        $positif \leftarrow \top$
      sinon si $\frac{|\{s' | s'(v) = v_i \wedge \exists (s, s') \in \mathcal{T}_{obs}\}|}{\{s' | \exists (s, s') \in \mathcal{T}_{obs}\}} \geq \tau$ **alors**
        $positif \leftarrow \top$
      fin
    fin
  fin
  si $positif = \top$ **alors**
    $\mathcal{S}^+_{v_i} \leftarrow \mathcal{S}^+_{v_i} \cup \{s\}$
  sinon
    $\mathcal{S}^-_{v_i} \leftarrow \mathcal{S}^-_{v_i} \cup \{s\}$
  fin
fin
retourner $\mathcal{S}^+_{v_i}$, $\mathcal{S}^-_{v_i}$

Algorithme 5 : Formation d'instances *LOLH* à partir de transitions *single-cell*.

Échantillonnage et sélection d'un jeu de paramètres

Les paramètres introduits précédemment paraissent difficiles à établir a priori. Nous pouvons tout de même tenter d'évaluer l'intérêt d'un jeu de valeurs particulier en utilisant le critère de qualité d'instance *LOLH* défini en annexe B. Ce critère entre 0 et 1 permet de déterminer à quel point il est possible de trouver des atomes corrélés à $\mathcal{S}^+$ et $\mathcal{S}^-$ pour

l'inférence de règle d'une instance particulière. On part alors du principe que les valeurs de paramètres les plus intéressantes sont trouvées lorsque celles-ci permettent d'obtenir de bonnes corrélations pour les atomes. En plus de cette valeur de qualité, nous étudions la proportion d'exemples positifs et négatifs qui est également sensible à ces paramètres. Afin de sélectionner un jeu de paramètres adéquat, nous fixons un ensemble de valeurs à tester pour δ : $\delta \in \{\varepsilon, 0.2, 0.4, 0.6, 0.8\}$ et nous testons la formulation des ensembles $\mathcal{S}^+$ et $\mathcal{S}^-$ à partir des transitions pour chaque atome des données, avec toutes les combinaisons de valeurs de paramètres possibles. Cela donne donc 30 jeux de valeurs à tester. Nous avons effectué cette démarche grâce à un accès au supercalculateur de Centrale Nantes *LIGER*[10]. La totalité des tests a été effectuée en environ 5 heures de calcul.

À partir des résultats obtenus pour chacun des 30 jeux de valeurs testés, on peut visualiser les valeurs de qualité et de proportions entre $\mathcal{S}^+$ et $\mathcal{S}^-$ pour les instances formées à partir de chaque atome. Les figures en annexe G nous permettent d'identifier les jeux de valeurs les plus intéressants pour l'exploitation des transitions. On remarque par exemple que lorsque $\rho = 1$, ou que $\tau = \varepsilon$, il y a un fort déséquilibre entre le nombre d'exemples positifs et le nombre d'exemples négatifs. La répartition semble plus homogène lorsque $\rho = 0$, et se rapproche de ce qu'on peut observer lorsqu'on effectue le même calcul pour les instances de co-expression (voir figure B.3). Nous sélectionnons donc le jeu de paramètres $(\tau = 0.6, \rho = 0, \delta = 1)$ pour la suite de l'analyse.

4.3.2 Inférence d'un réseau dynamique

À partir des paramètres sélectionnés précédemment, nous pouvons inférer un réseau entre les atomes logiques en utilisant d'abord l'algorithme 5 afin de former les instances de classification pour *LOLH*, puis la procédure présentée dans la partie 4.2.2. Pour la suite, puisque ce réseau a pour but de modéliser des relations dynamiques entre les gènes, nous l'appelons "réseau dynamique" pour le distinguer des réseaux de co-expression inférés dans la partie 4.2.2. Le réseau inféré ici est en effet obtenu en calculant des atomes corrélés dans les transitions (entre l'état de départ et l'état d'arrivée des transitions) contrairement au réseau de co-expression qui est obtenu en calculant les atomes corrélés dans les mêmes cellules. Comme précédemment, dans cette partie ce réseau est inféré en excluant les gènes mitochondriaux et les gènes ribosomaux. Pour ce réseau, seuls les atomes variant dans au

10. Cet accès entre dans le cadre de l'appel à projet *GLICID* : https://supercomputing.ec-nantes.fr/apply/allocation-schemes.

moins 50 cellules sont considérés et les corrélations sont conservées uniquement pour les atomes ayant un score supérieur ou égal à 0.5.

Clustering du réseau dynamique

Comme pour le réseau de co-expression, il est possible d'appliquer un algorithme de clustering afin de détecter les atomes fortement corrélés. Bien que dans cette partie nous cherchons à interpréter des relations dynamiques entre les gènes, le clustering du réseau permet de déterminer sa structure globale et cela offre un point de comparaison avec le graphe de co-expression. Pour ce graphe, nous appliquons comme précédemment l'algorithme de clustering *Louvain* avec un paramètre égal à 1.0. Nous ne donnons pas entièrement les clusters d'atomes obtenus ici pour des raisons de clarté, mais ceux-ci peuvent être recalculés et consultés à partir de l'implémentation en ligne de *LOLH* (voir partie `Examples`). Pour simplifier l'analyse de ces clusters, et puisque certains d'entre-eux comportent très peu d'atomes, nous nous concentrons par la suite exclusivement sur ceux contenant au moins 20 atomes logiques. Les 12 clusters retenus avec cette sélection présentent des symétries, de manière similaire à ce qui a été observé sur les réseaux de co-expression. On peut ainsi identifier les paires de clusters suivantes : 0 et 6 ; 1 et 7 ; 2 et 8 ; 3 et 9 ; 4 et 11 ; 20 et 23. Nous nous concentrons pour la suite sur l'analyse des clusters contenant des atomes correspondant à des gènes exprimés, c'est-à-dire les clusters 6 (contenant 350 atomes), 7 (169 atomes), 8 (322 atomes), 9 (181 atomes), 11 (92 atomes) et 20 (26 atomes). Comme cela a déjà été montré, il est possible de visualiser l'erreur de couverture des clusters sur la représentation 2d des données, comme cela est proposé sur la figure 4.11.

De la même manière que dans la partie précédente, il est possible de proposer une interprétation des clusters en comparant les gènes contenus à la liste de biomarqueurs en annexe D, ainsi qu'en étudiant les cellules obtenant un bon score de couverture par rapport aux atomes présents dans ces clusters. Ainsi, le cluster 6 comporte des marqueurs de myéloïdes avec les atomes $LST1_{1/1}$, $FCN1_{1/1}$, $LYZ_{1/1}$, $S100A9_{1/1}$, $CD14_{1/1}$ et $LGALS2_{1/1}$ correspondant aux cellules *CD14*, ainsi que les atomes $FCGR3A_{1/1}$ et $CSF1R_{1/1}$ correspondant aux cellules *CD16*. On trouve également $CD3E_{0/2}$ et $IL32_{0/2}$, indiquant la non-présence de gènes exprimés dans les lymphocytes *T*. Le cluster 7 comporte des marqueurs de lymphocytes *B*, avec $HLA\text{-}DMA_{1/1}$, $HLA\text{-}DQA1_{1/1}$, $CD79A_{1/1}$, $CD74_{1/1}$, $MS4A1_{1/1}$, $CD79B_{1/1}$, $PAX5_{1/1}$, $CD37_{2/2}$, $VPREB3_{1/1}$, $BLK_{1/1}$, $SPIB_{1/1}$, $TCL1A_{1/1}$, $STAP1_{1/1}$ et

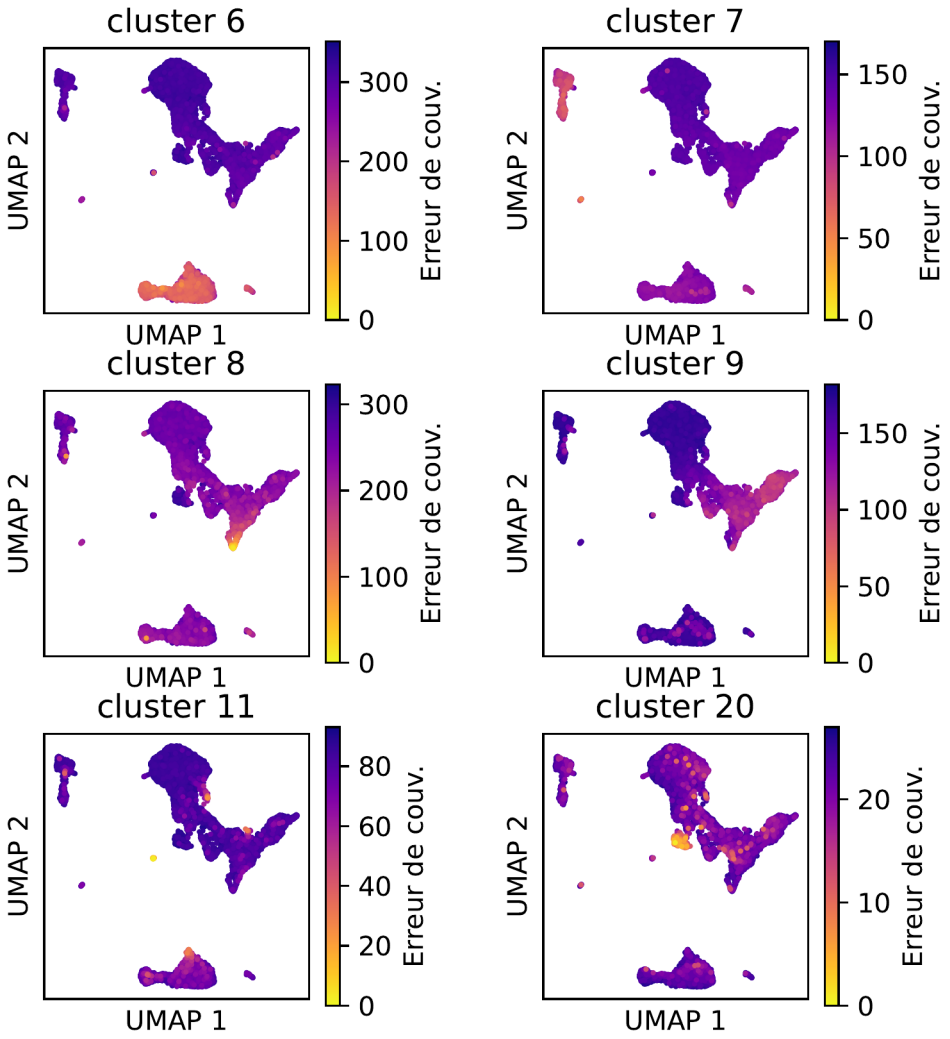

FIGURE 4.11 – Erreur de couverture des clusters du réseau dynamique sur l'ensemble des cellules, visualisée sur la représentation 2d des données.

$TNFRSF13B_{1/1}$. On retrouve également la non-expression de gènes associés à d'autres types, avec $S100A4_{0/2}$ (*cDc*) et $IL7R_{0/1}$ (*T*, *CD4*, *CD8*). Enfin, le cluster 9 comporte des marqueurs de lymphocytes *T* et de cellules *NK*. On retrouve en effet les atomes $GNLY_{1/1}$, $NKG7_{1/1}$, $PRF1_{1/1}$, $IL2RB_{1/1}$, $KLRC1_{1/1}$, $NCAM1_{1/1}$ (marqueurs des cellules *NK*), les atomes $CST7_{1/1}$, $KLRD1_{1/1}$, $CCL5_{1/1}$ (marqueurs des cellules *NK* et *CD8*), $GZMB_{1/1}$, $GZMH_{1/1}$, $TBX21_{1/1}$ (marqueurs des types *NK*, *CD4* et *CD8*), $CD8A_{1/1}$, $CD8B_{1/1}$, $KLRG1_{1/1}$ (marqueurs des cellules *CD8*), et $CD3D_{1/1}$, $IL32_{2/2}$ (marqueurs des cellules *T*, *CD4* et *CD8*).

La figure 4.11 montre par ailleurs que le cluster 11, contenant 92 atomes, se comporte de manière similaire au cluster 5 du graphe de co-expression inféré à partir du sous-jeu de données (cellules myéloïdes) dans la partie 4.2.2 (voir figure 4.9). Cela est confirmé par la présence du gène *PPBP*, précédemment identifié. On remarque également que ce cluster 11 contient les 13 atomes présents dans le cluster 5 du graphe de co-expression du sous-jeu de données. Enfin, si on compare la figure 4.11 avec la figure 4.2 en partie 4.1.3, les clusters 8 et 20 semblent correspondre aux lymphocytes *T*. Ces derniers clusters ne contiennent par ailleurs pas de marqueurs de la liste en annexe D.

Comparaison avec le réseau de co-expression

Dans cette partie, nous comparons plus en détail le réseau dynamique inféré précédemment avec le réseau de co-expression de la partie 4.2.2, afin de déterminer si l'utilisation des transitions extraites à partir d'un graphe de voisinage permet de calculer de nouvelles relations entre les atomes logiques.

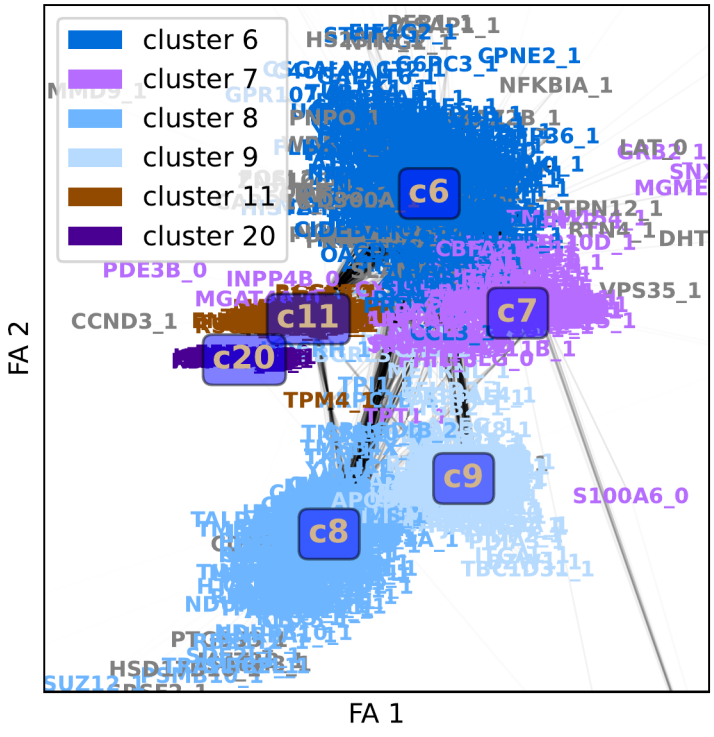

FIGURE 4.12 – Représentation 2d du réseau dynamique avec l'algorithme *Force Atlas 2*.

Dans un premier temps, il est possible de comparer les clusters obtenus pour le réseau dynamique avec ceux obtenus à partir du réseau de co-expression. On remarque que certains clusters comportent de nombreux atomes en commun. On peut ainsi associer les clusters suivants, entre le réseau dynamique et le réseau de co-expression respectivement : 0 et 4 (193 atomes en commun) ; 1 et 3 (48 atomes en commun) ; 3 et 0 (56 atomes en commun) ; 6 et 5 (203 atomes en commun) ; 7 et 2 (51 atomes en commun) ; 9 et 1 (58 atomes en commun). Ces associations confortent l'identification des types cellulaires effectuée précédemment sur les clusters du réseau dynamique, et nous pouvons aussi le confirmer visuellement à partir de l'erreur de couverture des clusters sur les cellules, en comparant les figures 4.6 et 4.11. On remarque par ailleurs que les clusters du réseau dynamique contiennent un nombre d'atomes plus important que leurs équivalents du réseau de co-expression.

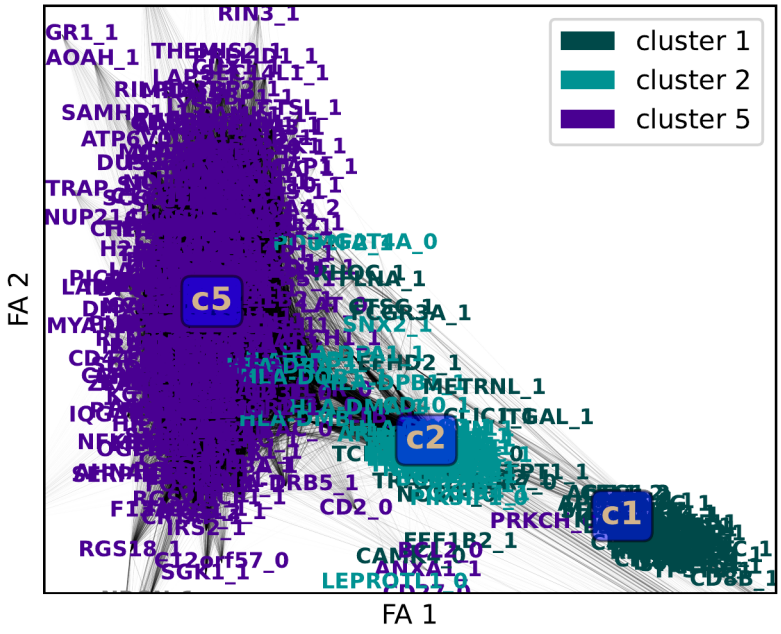

FIGURE 4.13 – Représentation 2d du graphe de co-expression avec l'algorithme *Force Atlas 2*.

Dans un second temps, il est possible de visualiser les réseaux de co-expression et dynamique à l'aide d'algorithmes de projection 2d. Nous utilisons ici l'algorithme *Force Atlas 2* qui a notamment déjà été utilisé dans le cadre de l'analyse de données *single-cell* (LEGOUX

et al., 2019). Pour cela, nous avons choisi l'implémentation *fa2* [11] (Jacomy et al., 2014) car celle-ci est réalisée dans le langage python, déjà utilisé pour le clustering des réseaux. Afin de simplifier l'analyse, nous nous concentrons uniquement sur les parties des réseaux dans lesquelles les atomes logiques correspondent à des gènes exprimés, ce qui correspond aux clusters 6, 7, 8, 9, 11 et 20 pour le réseau dynamique, et aux clusters 1, 2 et 5 pour le réseau de co-expression. Le réseau dynamique peut être visualisé sur la figure 4.12 et le réseau de co-expression est présent sur la figure 4.13. On remarque que les clusters 6, 7 et 9 du réseau dynamique sont agencés de manière similaire à leurs équivalents 5, 2 et 1 dans le réseau de co-expression. On distingue par ailleurs nettement les clusters 8, 11 et 20 du réseau dynamique qui ne semblent pas avoir d'équivalents dans le réseau de co-expression.

Ces observations nous montrent que le réseau dynamique et le réseau de co-expression ont certains clusters en commun. Le réseau dynamique présente par ailleurs de nouveaux clusters, qui peuvent être également associés à des caractéristiques de types cellulaires. Bien que ces clusters n'aient pas été obtenus dans la partie précédente, il a été remarqué a posteriori que ceux-ci apparaissent lors de l'inférence de réseaux de co-expression dans le cas où les gènes mitochondriaux et ribosomaux ne sont pas exclus. Le réseau présenté dans cette partie ne semble ainsi pas capturer de nouvelles relations entre les atomes. On peut cependant noter que l'exclusion des gènes mitochondriaux et ribosomaux constitue un élément de comparaison intéressant entre les deux approches. En effet, le cluster 11 de ce nouveau réseau présente un équivalent uniquement pour le graphe de co-expression spécifique aux cellules myéloïdes, et les clusters 8 et 20 n'ont pas d'équivalents dans les réseaux de co-expression (avec exclusion des gènes mitochondriaux et ribosomaux). La méthode présentée dans cette partie semble ainsi permettre une interprétation légèrement différente des corrélation entre les gènes.

4.4 Discussion

Dans ce chapitre, nous avons proposé une démarche pour la construction de réseaux génétiques à partir d'un jeu de données *single-cell*. Notre méthode a ainsi consisté à appliquer l'algorithme *LOLH* sur un jeu de données afin d'inférer des règles logiques entre les gènes. Puisque l'inférence d'un modèle qualitatif à partir de données expérimentales constitue une étape difficile, nous avons proposé une approche basée sur la construction

11. Le package est disponible à l'adresse https://github.com/bhargavchippada/forceatlas2.git.

et sur l'analyse de réseaux qui a été inspirée par les outils utilisés au niveau des cellules dans l'analyse de données *single-cell*. Cela nous a ainsi permis de valider notre méthode algorithmique, et d'aller plus loin dans l'exploitation les données *single-cell*. En particulier, l'application de *LOLH* sur un problème de classification a montré que celui-ci était en mesure de sélectionner les gènes pertinents pour expliquer la différence entre deux groupes de cellules. Cela a également permis de vérifier que l'utilisation des données discrétisées ne limite pas l'exploitation de relations entre les gènes.

La modélisation de la co-expression se révèle par ailleurs particulièrement intéressante du point de vue de l'exploitation des données. En effet, par rapport aux méthodes d'analyse de données *single-cell* couramment utilisées, la modélisation basée sur *LOLH* consiste à effectuer une analyse sur des réseaux de gènes, au lieu de considérer des graphes de cellules. Cet aspect a notamment permis de mettre en évidence, sur les données que nous avons traitées dans ce chapitre, certaines similarités entre cellules qui n'avaient pas été établies auparavant. Notre approche de modélisation, en contraste avec l'utilisation des méthodes de clustering sur les cellules, permet en effet de caractériser une même cellule par plusieurs groupes distincts de gènes d'intérêt. On peut par exemple imaginer une première caractérisation par rapport au type de la cellule, et une deuxième caractérisation par rapport à un stimuli extérieur.

Enfin, l'approche proposée pour la modélisation dynamique des données *single-cell* a permis de mettre en pratique notre algorithme *LOLH* dans le but de modéliser la régulation génétique. Bien que les résultats obtenus n'indiquent pas si les relations détectées concernent l'aspect dynamique du système, cette approche montre toutefois des différences intéressantes avec la modélisation de la co-expression, et de nombreux paramètres restent à explorer concernant cette direction. Plusieurs améliorations sont enfin envisageables, notamment sur la méthode d'extraction de transitions à partir des données. Il serait ainsi intéressant de comparer l'approche utilisée dans ce chapitre avec l'exploitation de données dynamiques obtenues avec les méthodes *RNA-velocity*, mentionnées dans la sous-partie 4.1.2. Enfin, afin de concrétiser notre approche de modélisation, nous voudrions proposer l'inférence d'un modèle qualitatif complet pouvant être simulé. Plusieurs étapes sont cependant requises pour atteindre cet objectif, comprenant l'extension de *LOLH* pour inférer de multiples règles ainsi que l'élaboration d'une sémantique adaptée pour la simulation des données *single-cell*.

Conclusion et perspectives

Conclusion

Ce travail de thèse a permis d'explorer une approche de modélisation qualitative des systèmes biologiques sous l'angle de la programmation logique inductive, avec son application sur des données réelles obtenues à partir des technologies de séquençage *single-cell*. L'interprétation d'un système biologique comme un système de transitions apporte de nouvelles perspectives pour son étude. Cependant, une double difficulté se présente pour une telle approche de modélisation. Pour commencer, du point de vue de l'inférence de modèles, le choix du formalisme ainsi que celui de l'algorithme d'inférence ont une grande importance sur la pertinence des modèles calculés. Des méthodes de validation rigoureuses sont donc nécessaires pour évaluer les différentes approches existantes. La deuxième difficulté réside dans la résolution des problèmes de calcul rencontrés lors de l'inférence et de l'analyse de modèles, puisque ces problèmes peuvent avoir une complexité algorithmique élevée.

Le chapitre 2, consacré à la résolution du problème d'accessibilité dans les réseaux d'automates, illustre les multiples perspectives se présentant pour la résolution d'un problème complexe et formalisé. La combinaison des méthodes de *model checking* et d'analyse statique est ainsi intéressante pour traiter la résolution théorique du problème d'accessibilité. Nous avons montré que l'implémentation de la Borne de Causalité Locale permet d'améliorer le pouvoir de résolution de ce problème. Cette méthode apporte également des perspectives d'améliorations, par exemple en étendant le calcul de la borne pour les Graphes de Causalité Locale cycliques. Ces éléments suggèrent que la compréhension théorique d'un formalisme de modélisation peut se révéler utile pour traiter différents problèmes d'analyse. La combinaison de plusieurs approches de résolution différentes semble également être une stratégie intéressante afin de tirer profit des avantages de chacune dans une même méthode.

Concernant le problème de l'inférence de modèles qualitatifs, nous avons proposé dans le chapitre 3 de nouvelles hypothèses de modélisation pour l'application du framework *LFIT* sur des données expérimentales de grandes dimensions, et possédant un aspect bruité. Une fois le problème formalisé sous ces hypothèses, nous avons proposé un nouvel algorithme, *LOLH*, basé sur la résolution d'un problème d'optimisation combinatoire, et possédant une faible complexité algorithmique. Nous avons montré la pertinence de *LOLH* pour l'inférence logique grâce à une comparaison avec des algorithmes existants sur des données artificielles. L'application de l'optimisation combinatoire à un problème d'inférence logique se révèle ainsi intéressante car de multiples éléments peuvent être empruntés au domaine de la Recherche Opérationnelle dans le but d'améliorer la résolution du problème. Notre implémentation de *LOLH*, disponible en ligne, permet facilement la mise en pratique de cette méthode sur différentes données.

Finalement, le chapitre 4 a démontré l'intérêt de *LOLH* sur un jeu de données *single-cell* réel dans le cadre d'une collaboration avec des chercheurs de l'institut Imagine. Nous avons dans un premier temps validé l'intérêt concret de notre algorithme avec une application sur des problèmes indépendants de l'aspect dynamique, en confrontant les résultats obtenus avec des connaissances biologiques déjà établies. Nous avons ensuite proposé une approche afin de traiter l'inférence d'un modèle dynamique à partir d'informations pseudo-temporelles extraites d'un jeu de données *single-cell* (en utilisant un graphe de voisinage). Cette approche, basée sur l'analyse de réseaux, a permis d'inférer un réseau qui diffère de celui obtenu via la modélisation de la co-expression. Notre méthode offre ainsi des perspectives pour poursuivre le développement de la modélisation dynamique des réseaux de régulation génétique à partir des données *single-cell*. Notre implémentation en ligne, qui permet notamment de reproduire toutes les analyses présentes dans le chapitre 4, s'inscrit également dans un effort de développement de ces méthodes de modélisation.

Pour conclure de manière plus générale, les difficultés rencontrées dans la modélisation qualitative des systèmes biologiques témoignent de la complexité de l'élaboration d'une approche de modélisation sur des données réelles. Au-delà des difficultés algorithmiques présentées, la formalisation de problèmes informatiques pertinents semble nécessiter un dialogue entre les chercheurs des différentes disciplines, nécessitant plusieurs allers-retours. Les progrès effectués dans les différentes disciplines de l'informatique sont bénéfiques en ce sens car ils permettent formaliser plus facilement certains problèmes réels. En particulier,

le domaine de l'apprentissage automatique apporte un autre regard sur la composante "donnée" présente en informatique. Celle-ci encourage en effet le développement de spécifications non formelles pour les problèmes réels, en comparaison avec les modélisations plus classiques où les spécifications formelles sont limitées en expressivité. Cette vision automatique facilite ainsi la modélisation à partir de données réelles, et nous pouvons imaginer que le renforcement des liens entre les différentes disciplines permettra à l'avenir d'améliorer davantage la qualité des modèles.

Perspectives

De nombreuses perspectives sont envisagées à partir de ces travaux. En ce qui concerne l'analyse des modèles qualitatifs, plusieurs extensions peuvent être proposées pour notre approche de résolution de l'accessibilité présentée dans le chapitre 2. La combinaison de l'analyse statique avec le *Bounded Model Checking* a permis de renforcer le lien établi entre le Graphe de Causalité Locale et les solutions du problème d'accessibilité. Afin de mieux traiter les cas où les Graphes de Causalité Locale sont cycliques, il peut être intéressant de développer des propriétés permettant de mieux caractériser la forme des arbres des objectifs correspondant à des solutions minimales. Ces propriétés peuvent alors être utilisées pour supprimer certains sommets des Graphes de Causalité Locale qui ne peuvent correspondre à des sommets dans les arbres des objectifs. Une suppression de sommets des GCL vise ainsi à faire disparaître les cycles sous certaines conditions. Dans un second temps, il serait intéressant de réutiliser l'approche de résolution en analyse statique afin de l'étendre pour traiter l'analyse des programmes logiques inférés dans le cadre du framework *LFIT*. En effet, la proximité entre le formalisme des Réseaux d'Automates et celui des Programmes Logiques incite à adopter une approche similaire pour développer des conditions nécessaires sur le problème d'accessibilité. De telles conditions nécessaires peuvent contribuer à simplifier l'analyse de programmes logiques, en les combinant par exemple à une approche de *Bounded Model Checking*.

Concernant l'inférence des modèles qualitatifs, plusieurs pistes d'améliorations ont été proposées pour l'algorithme *LOLH* dans le chapitre 3. Il serait ainsi intéressant d'étendre la formulation proposée afin d'inférer plusieurs règles logiques pour une même instance de classification. On peut en effet imaginer que l'inférence de multiples règles logiques permette d'obtenir un modèle plus complexe et plus fidèle au système étudié. Il peut éga-

lement être intéressant d'explorer plus en détail la formulation biobjectif du problème déjà proposée, et notamment étudier son intérêt dans un contexte appliqué où les exemples positifs et négatifs ont des significations concrètes. L'aspect dynamique que l'algorithme *LOLH* tente de capturer mérite également d'être étudié davantage. L'application de *LOLH* sur les données *single-cell* constitue en ce sens une première étape, mais plusieurs questions restent en suspens. Tout d'abord, l'extraction de transitions à partir d'un jeu de données *single-cell* est une étape cruciale pour le succès de la modélisation. Plusieurs alternatives à l'utilisation du graphe de voisinage sont envisageables. Par exemple, il peut s'avérer intéressant de combiner celui-ci avec des méthodes d'inférence pseudo-temporelles, afin d'obtenir un graphe orienté. On peut imaginer que l'orientation du graphe de voisinage peut faciliter la modélisation de certains mécanismes comme celui de la différenciation cellulaire. Les méthodes *RNA-velocity* représentent également une opportunité très intéressante pour l'extraction de transitions. Celles-ci apportent en effet une vision plus réaliste du comportement dynamique des cellules, car elles sont basées sur une information supplémentaire, qui est le taux de variation de la quantité de transcrits pour un gène donné. L'utilisation de ces méthodes pose cependant plusieurs difficultés car, bien qu'elles facilitent l'interprétation dynamique des données, elles reposent sur la notion de dérivée qui est difficile à modéliser de manière discrète. Les technologies *single-cell* étant encore en plein développement, on peut aussi envisager que l'amélioration de celles-ci facilitera l'obtention d'informations dynamiques dans le futur. Enfin, concernant la modélisation qualitative, la méthode d'inférence de la dynamique que nous avons proposée dans le chapitre 4 se révèle encourageante, mais le modèle obtenu n'a pour le moment pas été utilisé pour analyser ou encore simuler des séquences d'évolution du système. En ce sens, nous avons abordé le fait que la sémantique synchrone semble plus adéquate. Cependant, d'autres questions se posent sur les comportements obtenus par simulation d'un programme logique inféré à partir de données *single-cell*. En effet, les règles logiques inférées reposent sur les observations disponibles, et celles-ci sont bruitées et limitées en nombres. Il est donc important de prendre en compte la nature expérimentale des données dans notre contexte de modélisation formelle basée sur la programmation logique. Par exemple, la notion d'erreur de couverture définie dans le chapitre 3 peut se révéler utile dans la définition de la sémantique puisque les règles logiques sont inférées dans le but de ne pas couvrir totalement les exemples positifs des atomes logiques. Plusieurs extensions peuvent être envisagées pour faciliter la simulation de modèles dans ce cadre, comme l'ajout de délais ou encore la modélisation de transitions stochastiques.

Pour terminer, nous rappelons qu'une des motivations du développement d'algorithmes d'inférence à partir de données expérimentales est la possibilité d'exploiter plus d'informations dans ces données, grâce à une modélisation plus complexe en comparaison avec les outils d'analyses existants. Nous envisageons ainsi l'application de notre méthode à des problèmes de recherche concrets, pour lesquels les données *single-cell* sont déjà produites et analysées via les outils existants. Par exemple, notre approche pourrait se révéler utile dans des contextes de recherche bio-médicale, dans le but de mettre en évidence des relations génétiques non visibles avec les méthodes actuelles.

Annexe A

Génération d'instance artificielle pour *LOLH*

Nous détaillons dans cette annexe la procédure utilisée pour générer aléatoirement un jeu de données contenant des exemples positifs et négatifs utilisables par *LFIT* (pour l'inférence de règles logiques). Un tel jeu de données artificiel a pour but de permettre l'évaluation de différents algorithmes d'inférence de règles sur une instance, avec des exemples positifs et négatifs déjà déterminés. La méthode de génération proposée ici a été mise en place afin d'obtenir des données similaires aux données *single-cell*, utilisées dans le chapitre 4. Nous étudions donc dans un premier temps la forme des données dans les instances obtenues à partir d'un jeu de données *single-cell* pour nous en inspirer. Nous proposons ensuite une procédure complète pour la génération de données artificielles. Il est possible de retrouver cette procédure dans l'implémentation en ligne de *LOLH*.

A.1 Analyse d'une instance de données *single-cell*

Nous étudions dans un premier temps une instance formée à partir du jeu de données *single-cell* de l'institut Imagine, utilisé dans le chapitre 4. Nous nous concentrons en particulier sur l'instance de classification des cellules *NK*, déjà étudiée dans le chapitre 4 (voir parties 4.1.3 en page 129 et 4.2.1 en page 129). Dans cette instance, on a $|\mathcal{S}^+_{NK}| = 829$ et $|\mathcal{S}^-_{NK}| = 8369$. Pour comprendre la manière dont les données sont distribuées, nous nous concentrons sur les erreurs positive et négative des atomes logiques (définition 3.7 en page 100). La figure A.1 permet de visualiser ces erreurs pour tous les atomes des données (chaque valeur discrète de chaque variable).

La droite noire de la figure A.1, passant par les points $(0, 0)$ et $(|\mathcal{S}^+_{NK}|, |\mathcal{S}^-_{NK}|)$, représente les atomes ayant un score de 0. On remarque que la plupart des atomes sont situés le long de cette droite, avec un score proche de 0. Ces atomes ne sont donc pas intéressants selon

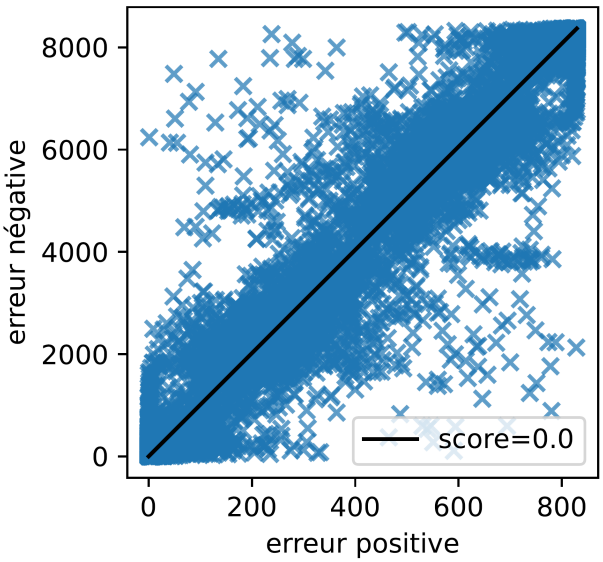

FIGURE A.1 – Erreurs positive et négative de tous les atomes pour la classification des cellules *NK*.

notre critère de sélection défini au chapitre 3, mais on observe tout de même un certain nombre d'atomes, en haut à gauche, qui peuvent être utilisés pour former une règle logique performante.

A.2 Génération aléatoire d'instances

En se basant sur les observations précédentes, nous pouvons tenter de générer un jeu de données dans lequel les scores des atomes ont une distribution semblable. Pour simplifier le problème, nous générons uniquement des variables binaires, ce qui résulte en deux atomes logiques par variable. Nous proposons ainsi de générer les données artificielles en deux étapes. Dans un premier temps, il est possible de choisir aléatoirement un score pour un atome donné, entre -1 et 1. À partir du score choisi, on peut générer l'erreur positive et l'erreur négative de l'atome de manière uniforme. Dans un second temps, nous pouvons utiliser les erreurs positive et négative des atomes pour générer aléatoirement les exemples positifs $\mathcal{S}^+$ et négatifs $\mathcal{S}^-$ constituant les données.

Génération des erreurs des atomes

Afin d'obtenir une densité d'atomes plus importante autour de 0, comme c'est le cas sur la figure A.1, nous choisissons de générer le score entre -1 et 1 en échantillonnant

une variable aléatoire suivant une loi de *Student*. On se place par la suite dans le cas de la génération des erreurs positive et négative pour un atome v_i donné. À partir de la définition du score d'un atome ($score(v_i) = \frac{erreur^-(v_i)}{|\mathcal{S}^-|} - \frac{erreur^+(v_i)}{|\mathcal{S}^+|}$), on peut déterminer une équation de droite des erreurs positive et négative d'un atome pour un score donné :

$$erreur^+(v_i) * |\mathcal{S}^-| - erreur^-(v_i) * |\mathcal{S}^+| + score(v_i) * |\mathcal{S}^+| * |\mathcal{S}^-| = 0 \tag{A.1}$$

Visuellement, sur une figure représentant les erreurs positive et négative des atomes, l'équation de droite pour un score donné correspond à une droite parallèle à la droite de score 0. La figure A.2 schématise la procédure employée pour générer l'erreur positive ($erreur^+(v_i)$) et l'erreur négative ($erreur^-(v_i)$) pour l'atome v_i. L'équation de droite correspondant au score de v_i peut être utilisée pour obtenir les coordonnées des points A et B aux extrémités du graphe (c'est-à-dire du carré $[0, |\mathcal{S}^+|] \times [0, |\mathcal{S}^-|]$). Enfin, les coordonnées de v_i, correspondant à son erreur positive et son erreur négative, peuvent être obtenues en choisissant aléatoirement un point sur la droite du score, entre A et B.

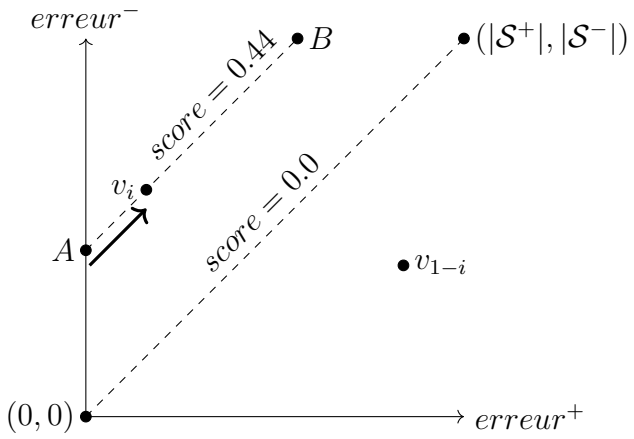

FIGURE A.2 – Détermination des erreurs positive et négative d'un atome à partir de son score et d'un paramètre aléatoire.

On peut remarquer que lorsque le score est positif, les points A et B sont situés à gauche et en haut du graphe, c'est-à-dire $x_A = 0$ et $y_B = |\mathcal{S}^-|$. Puisque A doit vérifier l'équation A.1, on a $0 * |\mathcal{S}^-| - y_A * |\mathcal{S}^+| + score(v_i) * |\mathcal{S}^+| * |\mathcal{S}^-| = 0$, ce qui donne $\boldsymbol{A = (0, score(v_i) * |\mathcal{S}^-|)}$. De manière similaire, on obtient $\boldsymbol{B = (|\mathcal{S}^+| * (1. - score(v_i)), |\mathcal{S}^-|)}$. Lorsque $score(v_i) < 0$, la point A est en bas du graphe ($y_A = 0$) et le point B est à droite ($x_B = |\mathcal{S}^+|$). On en déduit de l'équation A.1 que $\boldsymbol{A = (-score(v_i) * |\mathcal{S}^+|, 0)}$ et $\boldsymbol{B = (|\mathcal{S}^+|, |\mathcal{S}^-| * (score(v_i) + 1))}$.

À partir de A et B et d'une position aléatoire pos tirée uniformément entre 0 et 1, on peut calculer l'erreur positive et l'erreur négative de l'atome v_i de la manière suivante : $erreur^+(v_i) = (x_B - x_A) * pos, erreur^-(v_i) = (y_B - y_A) * pos$.
Puisque les atomes générés sont bi-valués, on peut également générer l'atome symétrique v_{1-i} avec $erreur^+(v_{1-i}) = |\mathcal{S}^+| - erreur^+(v_i)$ et $erreur^-(v_{1-i}) = |\mathcal{S}^-| - erreur^-(v_i)$ (la somme des erreurs doit être égale au nombre d'exemples positifs et négatifs respectivement). L'algorithme 6 résume la procédure complète pour générer les erreurs des atomes.

Données : $m_+, m_- \in \mathbb{N}$ le nombre d'exemples positifs et négatifs respectivement,
$\mathcal{V}$ un ensemble de variables, avec $|\mathcal{V}| = n \in \mathbb{N}$ le nombre de variables

$\mathcal{S}^+ \leftarrow \{s_i = \{0\}^n \mid i \in \{1, .., m_+\}\}$
$\mathcal{S}^- \leftarrow \{s_i = \{0\}^n \mid i \in \{1, .., m_-\}\}$
$\mathcal{A} \leftarrow \bigcup_{v \in \mathcal{V}} \{v_0, v_1\}$
pour $v \in \mathcal{V}$ **faire** `// calcul des erreurs des atomes`
  $score(v_0) \leftarrow échantilloner\_student()$
  $y \leftarrow échantilloner\_uniforme(0, 1)$
  si $score(v_0) \geq 0$ **alors**
    $A \leftarrow (0, score(v_0) * |\mathcal{S}^-|)$
    $B \leftarrow (|\mathcal{S}^+| * (1. - score(v_0)), |\mathcal{S}^-|)$
  sinon
    $A \leftarrow (-score(v_0) * |\mathcal{S}^+|, 0)$
    $B \leftarrow (|\mathcal{S}^+|, |\mathcal{S}^-| * (score(v_0) + 1))$
  fin
  $erreur^+(v_0) \leftarrow (x_B - x_A) * y$
  $erreur^-(v_0) \leftarrow (y_B - y_A) * y$
  $erreur^+(v_1) \leftarrow |\mathcal{S}^+| - erreur^+(v_0)$
  $erreur^-(v_1) \leftarrow |\mathcal{S}^-| - erreur^-(v_0)$
  $score(v_1) \leftarrow erreur^-(v_1)/|\mathcal{S}^-|/erreur^+(v_1)/|\mathcal{S}^+|$
fin

Algorithme 6 : Génération aléatoire des erreurs positive et négative des atomes.

Génération des exemples positifs et négatifs

Une fois que les erreurs des atomes sont générées, il reste à générer les ensembles d'états discrets $\mathcal{S}^+$ et $\mathcal{S}^-$ qui forment les données. Puisque nous ne faisons aucune hypothèse particulière concernant la répartition des erreurs positives et négatives dans les exemples

positifs et négatifs, nous choisissons de répartir les valeurs discrètes 0 et 1 des différentes variables de manière uniforme.

A.3 Analyse du résultat

Nous pouvons analyser le résultat de la génération en utilisant les méthodes de réduction de dimensionnalité mentionnées dans le chapitre 4. Nous utilisons les méthodes *PCA* (ACP) et *UMAP* pour visualiser les données en 2 dimensions. La figure A.3 permet de visualiser les projections obtenues, avec identification des exemples positifs et négatifs à partir de couleurs. La projection avec la méthode *UMAP* a été obtenue à partir des 10 premières composantes d'une projection *PCA*, en utilisant les paramètres suivants : $min\_dist = 0.3$, $n\_neighbors = 50$, $spread = 1.0$. On remarque sur la figure une très nette séparation entre les exemples positifs et négatifs, ce qui justifie les performances de l'algorithme *LOLH* obtenues dans le chapitre 3 pour la classification de ces données.

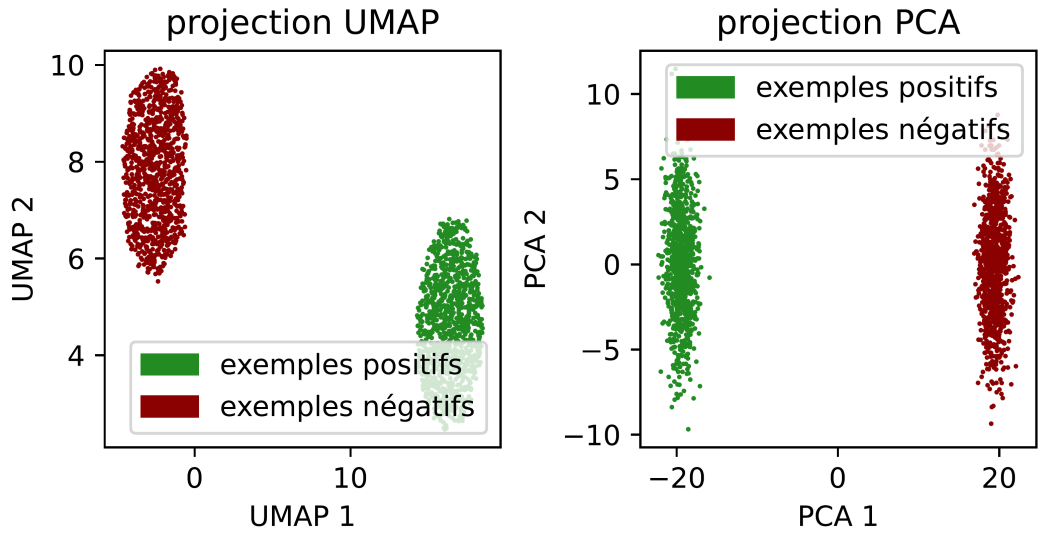

FIGURE A.3 – Réduction de dimensionnalité avec les méthodes de PCA et UMAP.

Annexe B

Critère de qualité d'une instance *LOLH*

Dans cette annexe, nous proposons un critère pour mesurer la qualité d'une instance *LOLH*, définie à partir d'ensembles d'exemples positifs $\mathcal{S}^+$ et négatifs $\mathcal{S}^-$. Il peut en effet être utile d'évaluer la "difficulté" d'une instance de classification donnée, c'est-à-dire à quel point il est difficile d'inférer un classifieur pour cette instance. Une telle mesure de qualité est notamment utile pour déterminer un jeu de paramètres intéressant pour l'extraction de données, comme cela est montré dans le chapitre 4.

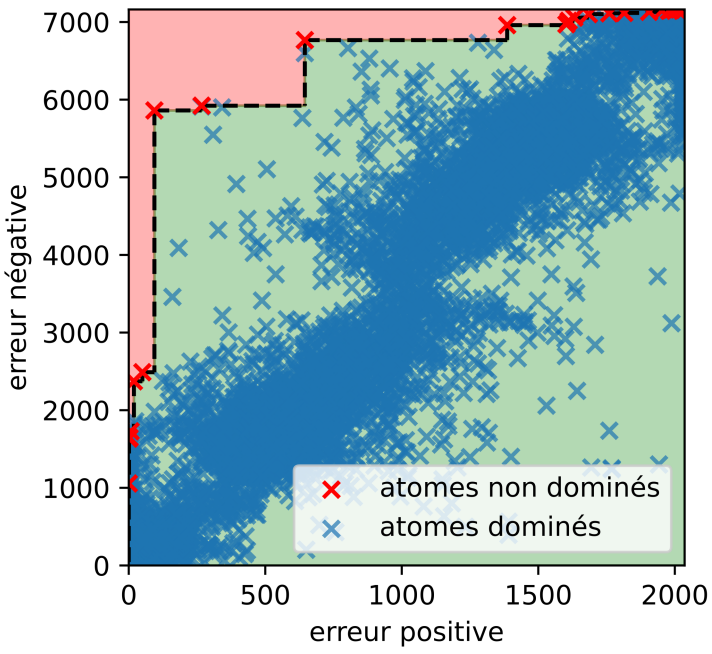

Figure B.1 – Erreurs positives et négatives de tous les atomes de l'instance de classification des cellules *CD8*. Les atomes en rouge sont ceux qui ne sont dominés par aucun autre atome.

Bien que l'on pourrait utiliser directement le score d'une règle optimale inférée avec *LOLH* pour établir ce critère, ce dernier indicateur a pour défaut d'être dépendant de l'algorithme utilisé et en particulier de ses paramètres (le seuil de sélection sur le score des atomes dans le cas de *LOLH*). Pour remédier à cela, on peut alors tenter d'établir un critère se basant uniquement sur les données et ne dépendant donc d'aucun paramètre. Nous nous basons ainsi sur les erreurs positives et négatives des atomes logiques (définition 3.7) pour l'instance considérée. Nous illustrons dans cette annexe la mise en place de ce critère sur des instances de classification de type cellulaire à partir du jeu de données de l'institut Imagine présenté en partie 4.1.3 (page 129). Ces instances sont formées de manière similaire à ce qui est proposé dans la partie 4.2.1 du chapitre 4. La figure B.1 montre ainsi les erreurs positives et négatives de l'ensemble des atomes pour l'instance de classification des cellules *CD8* et la figure B.2 illustre les erreurs des atomes pour les instances de classification des cellules *NK* et *CD4*.

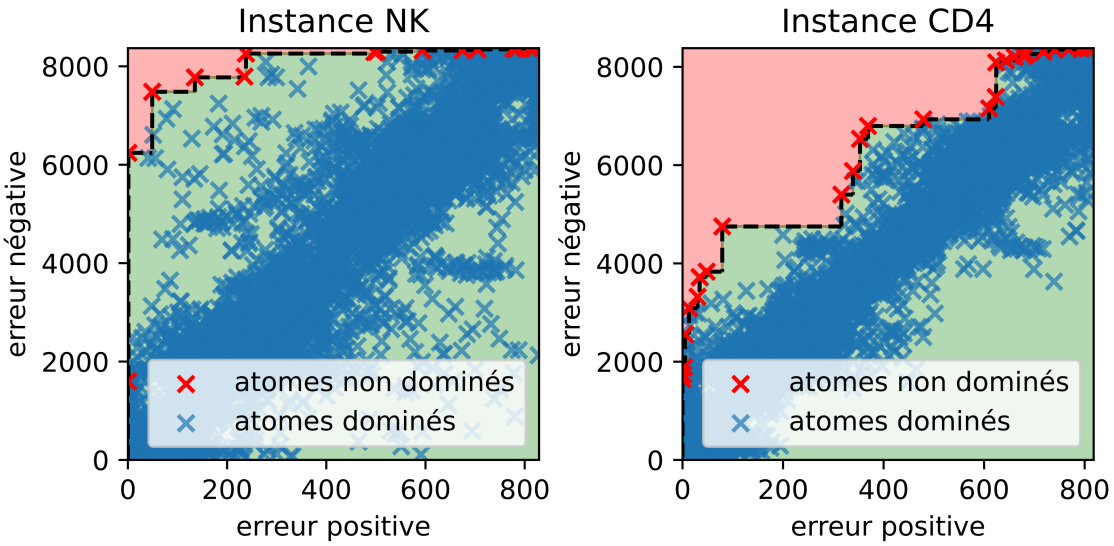

FIGURE B.2 – Erreurs positive et négatives de tous les atomes pour l'instance de classification des cellules *NK* à gauche et des cellules *CD4* à droite.

Intuitivement, on peut penser qu'une instance de classification *LOLH* est facile à traiter lorsqu'il est possible de trouver des atomes logiques fortement corrélés avec la caractéristique à classifier. La présence d'atomes logiques ayant une faible erreur positive et une forte erreur négative indique alors que l'instance est "facile" à résoudre (ces atomes correspondent en effet à ceux sélectionnés par *LOLH* en priorité). Nous formalisons ceci

grâce à la notion de dominance utilisée en optimisation multiobjectif[1]. On considère ainsi qu'un atome x_i domine un atome y_j si et seulement si :
$(erreur^+(x_i) < erreur^+(y_j) \wedge erreur^-(x_i) > erreur^-(y_j))$.
À partir de cette définition, on peut calculer facilement l'ensemble des atomes non dominés en utilisant un algorithme de tri. Les atomes non dominés sont représentés en rouge sur les figures B.1 et B.2. L'ensemble des atomes non dominés d'une instance permet alors de partitionner le plan des erreurs des atomes en deux sous-ensembles : un ensemble de points dominés, représenté en vert sur les figures et un ensemble de points non dominés représenté en rouge. L'évaluation de la qualité d'une instance peut alors se faire grâce à un calcul géométrique notamment utilisé en optimisation multiobjectif. Il est en effet possible de facilement calculer le rapport entre l'aire des points dominés et l'air total du plan des erreurs des atomes. Dans la classification des cellules *CD8*, on obtient un rapport de 0.896, ce qui est cohérent avec ce que l'on peut observer sur la figure B.1. On obtient par ailleurs un rapport de 0.958 pour la classification des cellules *NK*, ainsi qu'un rapport de 0.739 pour la classification des cellules *CD4*. Ces valeurs sont cohérentes avec ce que l'on peut observer sur la figure B.2, où les instances semblent respectivement plus faciles et plus difficiles que l'instance des cellules *CD8*.

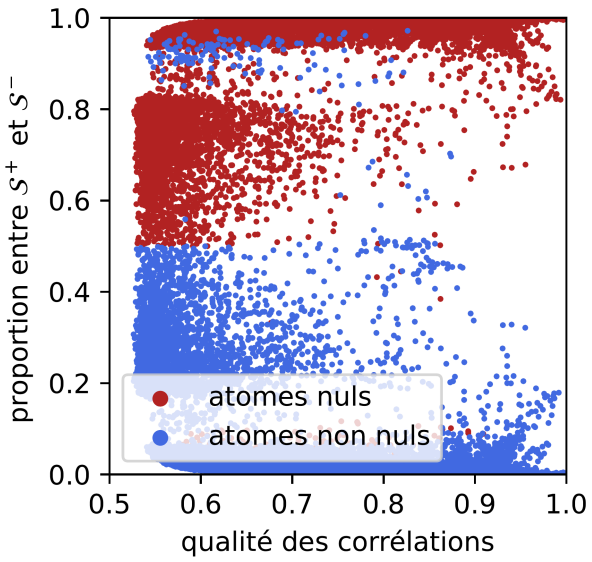

FIGURE B.3 – Qualité des corrélations obtenues par rapport à la proportion entre les exemples positifs et les exemples négatifs pour chaque instance *LOLH* créée à partir de la co-expression des atomes logiques.

1. Toutefois, en recherche opérationnelle cette notion est définie dans le cas des solutions d'un problème d'optimisation. Dans notre cas, nous l'utilisons directement sur les variables discrètes (atomes).

Le calcul de ce critère de qualité a été implémenté dans la version de *LOLH* disponible en ligne. Cette implémentation se distingue en deux étapes : une première étape consistant à calculer les points non dominés et une seconde étape calculant le rapport entre l'aire des points dominés et l'aire totale. Dans le cas de l'inférence de réseaux de co-expression proposée dans la partie 4.2.2 du chapitre 4, il est possible d'utiliser ce critère pour déterminer si des corrélations sont effectivement présentes entre les atomes. La figure B.3 permet de visualiser les qualités de chaque atome modélisé dans la co-expression. Ce graphique présente la valeur de qualité des atomes en abscisse, ainsi que la proportion entre les exemples positifs et négatifs pour l'instance des atomes en ordonnée. On remarque qu'une majorité d'atomes ont peu de corrélations, ce qui explique le faible nombre d'atomes dans les clusters de co-expression obtenus par rapport au nombre total de gènes.

Annexe C

Informations supplémentaires sur les données

C.1 Acquisition des données

Le jeu de données utilisé pour l'application de *LOLH* dans le chapitre 4 est une matrice *single-cell* ARN composée de 9198 cellules mononucléées sanguines périphériques (pbmc : *T*, *B*, *NK* et myéloïdes) extraites à partir de donneurs sains, fournie et prétraitée par l'institut Imagine. Les cellules étiquetées C26 (5396 cellules) proviennent d'une femme de 30 ans et celles étiquetées C27 come proviennent d'un homme de 53 ans (3802 cellules). Les cellules ont été isolées du sang à l'aide de *ficoll*. Les échantillons ont été séquencés avec un protocole chimique standard 3' v3 par *10x genomics*. *Cellranger v4.0.0* a été utilisé pour le traitement et les séquences ont été alignées à *ensembl GRCg38 human genome* (*GRCg38_r98-ensembl_Sept2019*). Les indicateurs de contrôle de qualité (*QC*) ont été calculés sur la matrice d'expression générée par *cellranger* (*filtered_feature_bc_matrix*). Les cellules avec moins de 3 gènes par cellules, moins de 500 molécules par cellule et moins de 20% de gènes mitochondriaux ont été écartées des données.

C.2 Prétraitement avec *Seurat*

Le prétraitement a été effectué avec le package R Seurat [1], comprenant l'intégration des échantillons, la normalisation des données et le *scaling*, la réduction de dimensionnalité et le clustering. La méthode *SCTransform* (Hafemeister et al., 2019) a été adoptée pour les étapes de normalisation et de *scaling*. Les clusters de cellules ont été manuellement annotés à l'aide de marqueurs de types cellulaires connus (figure 4.2). La figure C.1 montre les types de cellules généraux identifiés à partir des données sur la représentation UMAP.

1. https://satijalab.org/seurat/

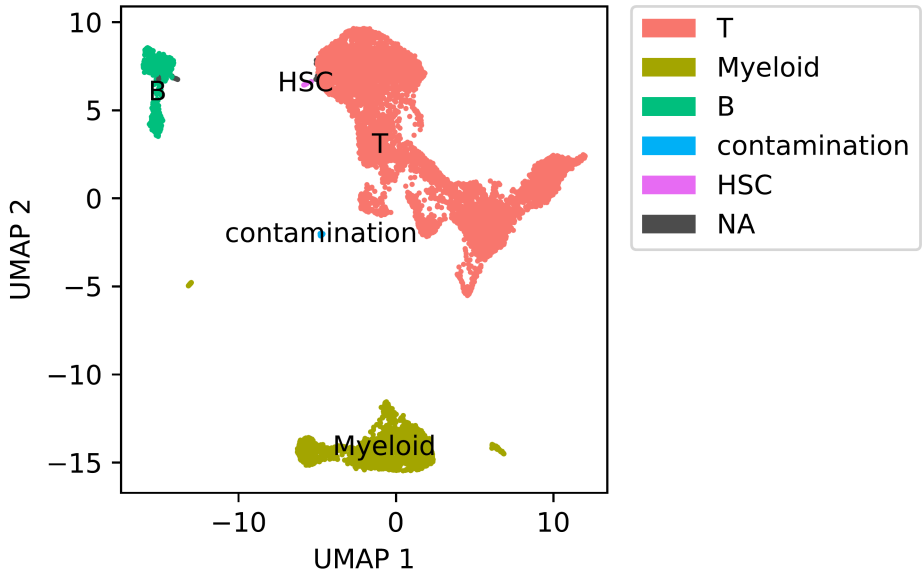

FIGURE C.1 – Représentation 2d calculée par la méthode *UMAP* des cellules colorées selon leur type.

C.3 Discrétisation des données

L'application de notre méthode *LOLH* nécessite la discrétisation préalable des données. Une méthode simple pour la discrétisation consiste à binariser les données, en fonction de la présence ou de l'absence d'un gène dans une cellule donnée. Cette méthode n'est cependant pas très précise et implique une perte d'information conséquente.

Afin de réaliser une discrétisation plus fine, nous avons procédé de manière empirique en nous basant sur la distribution des valeurs (normalisées) de chaque gène à travers les données. Nous avons ainsi catégorisé les gènes en fonction de leur distribution de valeurs à partir de plusieurs indicateurs comme la moyenne, la médiane, ou encore la variance. Cette séparation a permis de décider le nombre de valeurs discrètes pour chaque variable. La détermination des seuils a ensuite été effectuée de manière semi-automatique, en combinant les indicateurs calculés avec certains seuils manuels. La figure C.2 illustre l'utilisation d'une représentation graphique des distributions pour faciliter la catégorisation des gènes. La discrétisation finale a résulté en 14030 gènes ayant 2 valeurs discrètes, 134 gènes ayant 3 valeurs discrètes et 42 gènes avec 4 valeurs. Le programme utilisé pour discrétiser les données est disponible dans l'implémentation en ligne de *LOLH*.

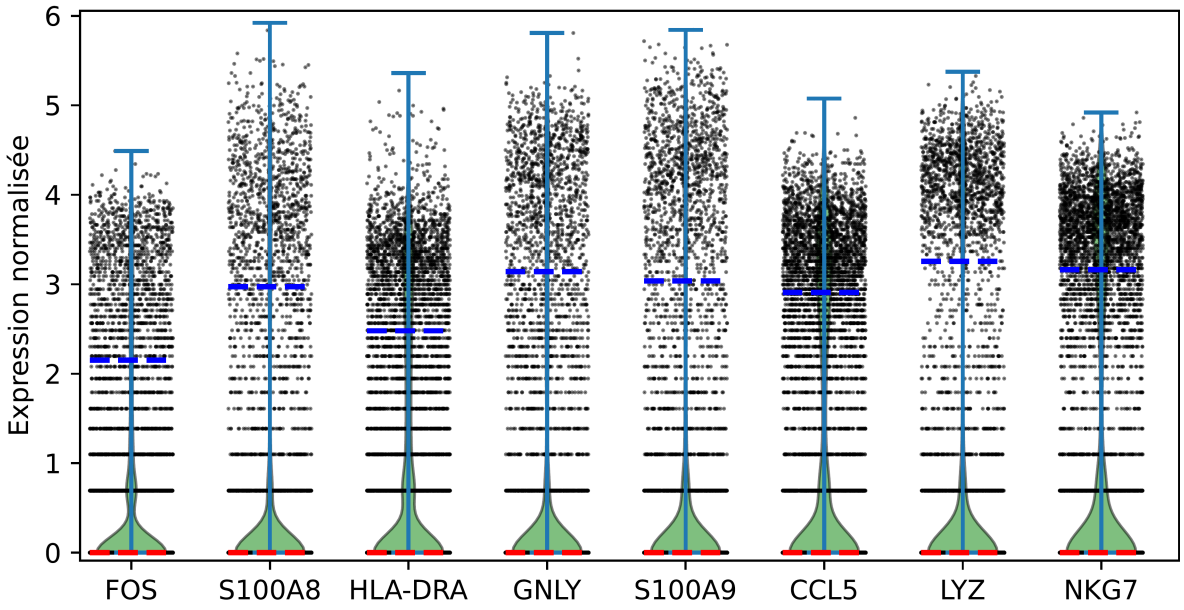

FIGURE C.2 – Représentation des distributions des valeurs de certains gènes dans les cellules.

Annexe D

Marqueurs biologiques connus

Gènes	B	NK	T	CD4	CD8	CD14	CD16	cDC
AIF1								✓
BLK	✓							
CCL2						✓		
CCL5		✓			✓			
CCR6	✓							
CD14						✓		
CD16							✓	
CD180	✓							
CD19	✓							
CD37	✓							
CD3D			✓	✓	✓			
CD3E			✓	✓	✓			
CD40LG				✓				
CD72	✓							
CD74	✓							✓
CD79A	✓							
CD79B	✓							
CD80	✓							
CD83								✓
CD8A					✓			
CD8B					✓			
CD96		✓	✓	✓	✓			
CSF1R							✓	
CST7		✓			✓			

Gènes	B	NK	T	CD4	CD8	CD14	CD16	cDC
CTLA4				✓				
CXCR5	✓							
FASLG		✓						
FCER1A								✓
FCER1G		✓						✓
FCGR3A		✓					✓	
FCN1						✓		✓
FCRL2	✓							
FLT3								✓
GNLY		✓						
GPR183								✓
GZMB		✓		✓	✓			
GZMH		✓		✓	✓			
HLA-DMA	✓							✓
HLA-DQA1	✓							✓
HTR3A	✓							
IL1R2								✓
IL2RB		✓						
IL32			✓	✓	✓			
IL6R				✓				
IL7R			✓	✓	✓			
ITGAX								✓
ITM2A			✓	✓	✓			
KIR2DL3		✓			✓			
KLRC1		✓						
KLRD1		✓			✓			
KLRG1					✓			
LGALS2						✓		✓
LGALS3								✓
LST1						✓		✓
LYZ						✓		
MS4A1	✓							

Gènes	B	NK	T	CD4	CD8	CD14	CD16	cDC
NCAM1		✓						
NKG7		✓						
PAX5	✓							
PIKFYVE	✓							
PNOC	✓							
PRF1		✓						
S100A4								✓
S100A9						✓		
SPIB	✓							
STAP1	✓							
TBX21		✓		✓	✓			
TCL1A	✓							
TCL1B	✓							
TLR7	✓							
TNFRSF13B	✓							
TYROBP								✓
VPREB3	✓							
ZBTB16		✓						

TABLE D.1 – Liste de marqueurs génétiques pour plusieurs types cellulaires. Cette liste a été proposée par l'institut Imagine.

Annexe E

Expression différentielle sur les données single-cell

E.1 Analyse d'expression différentielle sur les cellules *NK*

gène	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
GNLY	0.	4.48	$9.59e-001$	$1.55e-001$	0.
GZMB	0.	3.18	$8.99e-001$	$1.13e-001$	0.
KLRF1	0.	2.75	$8.56e-001$	$3.7e-002$	0.
SPON2	0.	2.54	$7.55e-001$	$1.45e-001$	0.
PRF1	0.	2.53	$9.59e-001$	$2.25e-001$	0.
CTSW	0.	2.48	$9.69e-001$	$3.5e-001$	0.
HOPX	0.	2.48	$8.67e-001$	$1.51e-001$	0.
PTGDS	$1.33e-155$	2.43	$1.39e-001$	$7.e-003$	$1.99e-151$
IL2RB	0.	2.38	$8.58e-001$	$1.44e-001$	0.
CLIC3	0.	2.37	$7.88e-001$	$1.3e-001$	0.
KLRB1	0.	2.36	$8.36e-001$	$1.71e-001$	0.
KLRD1	0.	2.27	$9.64e-001$	$2.21e-001$	0.
CMC1	0.	2.25	$8.01e-001$	$1.65e-001$	0.
NKG7	0.	2.11	1.	$3.42e-001$	0.
CD7	0.	2.09	$9.4e-001$	$5.13e-001$	0.
CST7	0.	2.06	$9.81e-001$	$3.07e-001$	0.
TRDC	0.	1.94	$7.74e-001$	$5.4e-002$	0.
FGFBP2	$1.38e-294$	1.92	$7.55e-001$	$1.99e-001$	$2.06e-290$
CD247	0.	1.91	$8.94e-001$	$4.46e-001$	0.

gène	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
GZMA	0.	1.78	$9.64e-001$	$2.96e-001$	0.
MAP3K8	$2.46e-252$	1.72	$7.23e-001$	$2.57e-001$	$3.68e-248$
KLRC1	0.	1.69	$3.64e-001$	$1.5e-002$	0.
CD63	0.	1.67	$8.73e-001$	$3.9e-001$	0.
IFITM2	$9.92e-273$	1.62	$9.73e-001$	$7.75e-001$	$1.48e-268$
AREG	$1.71e-279$	1.57	$4.25e-001$	$6.e-002$	$2.56e-275$
RUNX3	$8.1e-256$	1.54	$8.48e-001$	$3.94e-001$	$1.21e-251$
EFHD2	$7.93e-258$	1.53	$9.34e-001$	$4.81e-001$	$1.19e-253$
FCGR3A	0.	1.52	$7.78e-001$	$1.69e-001$	0.
GZMM	$3.77e-241$	1.51	$8.91e-001$	$4.52e-001$	$5.63e-237$
METRNL	$5.31e-290$	1.51	$6.83e-001$	$1.9e-001$	$7.94e-286$

TABLE E.1 – Résultat de l'analyse d'expression différentielle réalisée entre les cellules *NK* et les autres cellules du jeu de données. Le tableau présente les 30 gènes les plus différemment exprimés, triés selon la colonne *avg_log2FC*.

E.2 Analyse d'expression différentielle sur les clusters de co-expression

E.2.1 Analyse différentielle sur le cluster 6 du sous-graphe

gène	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
NKG7	$2.33e-036$	3.38	$9.02e-001$	$2.01e-001$	$3.49e-032$
CCL5	$1.16e-045$	3.30	$9.76e-001$	$1.98e-001$	$1.73e-041$
IL32	$1.40e-075$	3.09	$9.02e-001$	$8.e-002$	$2.1e-071$
GNLY	$2.17e-029$	2.92	$5.37e-001$	$7.1e-002$	$3.24e-025$
GZMA	$1.45e-107$	2.54	$8.29e-001$	$3.7e-002$	$2.17e-103$
GZMK	$1.71e-109$	2.23	$5.61e-001$	$1.2e-002$	$2.56e-105$
CD3E	$7.19e-080$	2.23	$9.02e-001$	$7.2e-002$	$1.07e-075$
IL7R	$1.61e-042$	2.14	$6.1e-001$	$6.1e-002$	$2.41e-038$
CST7	$2.9e-097$	2.13	$8.78e-001$	$5.1e-002$	$4.33e-093$
GZMH	$9.58e-090$	2.08	$6.59e-001$	$2.7e-002$	$1.43e-085$

gène	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
CD3G	$3.85e-086$	2.04	$8.78e-001$	$5.9e-002$	$5.75e-082$
ETS1	$4.35e-066$	1.87	$9.76e-001$	$1.05e-001$	$6.5e-062$
TRAC	$2.59e-083$	1.82	$8.54e-001$	$5.7e-002$	$3.87e-079$
TRBC2	$3.92e-071$	1.81	$8.78e-001$	$7.4e-002$	$5.87e-067$
SYNE2	$7.76e-075$	1.76	$8.29e-001$	$6.2e-002$	$1.16e-070$
CD3D	$1.07e-082$	1.75	$8.78e-001$	$6.1e-002$	$1.6e-078$
CD2	$6.65e-060$	1.73	$8.29e-001$	$8.2e-002$	$9.94e-056$
CD8A	$2.27e-118$	1.71	$7.32e-001$	$2.3e-002$	$3.4e-114$
CTSW	$1.43e-082$	1.58	$8.29e-001$	$5.4e-002$	$2.14e-078$
GZMM	$7.12e-104$	1.57	$8.78e-001$	$4.6e-002$	$1.06e-099$
...	...	...	...	...	...
LGALS1	$1.77e-006$	$-7.61e-001$	$9.27e-001$	$9.65e-001$	$2.65e-002$
FTL	$5.17e-012$	$-7.64e-001$	1.	$9.99e-001$	$7.73e-008$
FCN1	$3.59e-007$	$-7.64e-001$	$9.51e-001$	$9.62e-001$	$5.36e-003$
CD68	$3.28e-006$	$-7.7e-001$	$7.56e-001$	$8.72e-001$	$4.90e-002$
SPI1	$7.09e-007$	$-7.73e-001$	$9.02e-001$	$9.16e-001$	$1.06e-002$
AIF1	$2.48e-006$	$-7.73e-001$	$9.76e-001$	$9.79e-001$	$3.71e-002$
FCER1G	$6.92e-007$	$-7.97e-001$	$9.27e-001$	$9.69e-001$	$1.03e-002$
S100A6	$5.54e-011$	$-8.02e-001$	1.	$9.99e-001$	$8.28e-007$
TKT	$3.52e-009$	$-8.13e-001$	$9.76e-001$	$9.59e-001$	$5.26e-005$
TYROBP	$1.11e-011$	$-8.32e-001$	1.	$9.93e-001$	$1.66e-007$
VIM	$5.94e-009$	$-8.55e-001$	1.	$9.89e-001$	$8.89e-005$
CST3	$1.69e-009$	$-8.63e-001$	1.	$9.95e-001$	$2.52e-005$
LYZ	$7.81e-008$	-1.01	1.	$9.94e-001$	$1.17e-003$
LST1	$1.19e-009$	-1.01	$9.51e-001$	$9.82e-001$	$1.78e-005$
CSTA	$5.18e-009$	-1.1	$7.8e-001$	$9.03e-001$	$7.75e-005$

TABLE E.2 – Les 20 premiers et les 15 derniers gènes retournés par l'analyse d'expression différentielle entre cellules myéloïdes correspondant au cluster 6 du sous graphe et entre les autres cellules myéloïdes (partie 4.2.2 du chapitre 4).

E.2.2 Analyse différentielle sur le cluster 5 du sous-graphe

gène	p_val	avg_log2FC	pct.1	pct.2	p_val_adj
S100A9	$1.70e-040$	6.53	$9.83e-001$	$4.8e-002$	$2.55e-036$
LYZ	$1.93e-041$	6.37	1.	$7.6e-002$	$2.88e-037$
S100A8	$2.1e-035$	5.7	$9.17e-001$	$5.7e-002$	$3.14e-031$
FOS	$8.46e-038$	4.88	$9.67e-001$	$1.24e-001$	$1.26e-033$
CTSS	$3.83e-039$	4.36	1.	$4.1e-001$	$5.72e-035$
VCAN	$1.50e-036$	4.13	$9.09e-001$	$1.e-002$	$2.25e-032$
FCN1	$2.93e-041$	4.11	$9.83e-001$	$1.e-002$	$4.39e-037$
LST1	$3.12e-038$	3.58	$9.67e-001$	$1.33e-001$	$4.67e-034$
AIF1	$1.33e-038$	3.55	$9.75e-001$	$1.24e-001$	$1.98e-034$
FGL2	$5.87e-041$	3.52	$9.83e-001$	$2.9e-002$	$8.77e-037$
CYBB	$4.70e-039$	3.34	$9.83e-001$	$6.7e-002$	$7.03e-035$
S100A12	$2.36e-029$	3.27	$7.85e-001$	$1.e-002$	$3.53e-025$
NAMPT	$2.94e-036$	3.18	$9.34e-001$	$5.7e-002$	$4.39e-032$
DUSP1	$1.18e-036$	3.1	$9.92e-001$	$2.29e-001$	$1.77e-032$
SERPINA1	$8.09e-040$	3.08	$9.59e-001$	0.	$1.21e-035$
MNDA	$3.11e-038$	3.04	$9.42e-001$	$1.9e-002$	$4.65e-034$
PSAP	$5.54e-037$	2.77	1.	$4.29e-001$	$8.29e-033$
TYMP	$1.3e-034$	2.76	$9.5e-001$	$1.52e-001$	$1.94e-030$
TYROBP	$1.32e-034$	2.75	1.	$1.81e-001$	$1.97e-030$
KLF10	$2.25e-037$	2.75	$9.59e-001$	$8.6e-002$	$3.36e-033$
LGALS1	$2.19e-034$	2.72	$9.83e-001$	$1.52e-001$	$3.27e-030$
CSTA	$5.17e-036$	2.7	$9.01e-001$	$1.e-002$	$7.72e-032$
S100A6	$1.34e-036$	2.7	1.	$6.48e-001$	$2.00e-032$
SPI1	$1.5e-037$	2.66	$9.26e-001$	$1.e-002$	$2.24e-033$
EGR1	$6.94e-024$	2.62	$6.94e-001$	$1.9e-002$	$1.04e-019$
APLP2	$3.23e-037$	2.54	$9.67e-001$	$1.24e-001$	$4.83e-033$
CLEC7A	$1.93e-036$	2.53	$9.09e-001$	$1.e-002$	$2.89e-032$
CEBPD	$6.87e-035$	2.47	$9.26e-001$	$4.8e-002$	$1.03e-030$

TABLE E.3 – Les 28 gènes les plus différemment exprimés entre les myéloïdes correspondant au cluster 5 du sous-graphe, et entre les autres cellules du cluster 5 (partie 4.2.2).

Annexe F

Clusters d'atomes corrélés sur les données single-cell

F.1 Clusters sur le graphe initial

Cluster 0 (64 atomes) :

$GNLY_{0/1}$, $KLRB1_{0/1}$, $KLRF1_{0/1}$, $SYTL3_{0/1}$, $IL2RB_{0/1}$, $CTSW_{0/1}$, $GZMB_{0/1}$, $CST7_{0/1}$, $NKG7_{0/1}$, $GZMA_{0/1}$, $CCL5_{0/1}$, $KLRD1_{0/1}$, $GZMM_{0/1}$, $PRF1_{0/1}$, $FGFBP2_{0/1}$, $ABHD17A_{0/1}$, $MATK_{0/1}$, $SPON2_{0/1}$, $EFHD2_{0/1}$, $GZMH_{0/1}$, $APMAP_{0/1}$, $CD99_{0/1}$, $CLIC3_{0/1}$, $PYHIN1_{0/1}$, $CHST12_{0/1}$, $ARL4C_{0/1}$, $DUSP2_{0/1}$, $RUNX3_{0/1}$, $LITAF_{0/1}$, $HOPX_{0/1}$, $CD81_{0/1}$, $C12orf75_{0/1}$, $CLIC1_{0/1}$, $ACTN4_{0/1}$, $PLAAT4_{0/1}$, $SYNE2_{0/1}$, $ID2_{0/1}$, $METRNL_{0/1}$, $ITGAL_{0/1}$, $MAP3K8_{0/1}$, $SYNE1_{0/1}$, $CD8A_{0/1}$, $APOBEC3G_{0/1}$, $CD8B_{0/1}$, $IL2RG_{0/1}$, $SAMD3_{0/1}$, $PPP2R5C_{0/1}$, $LYAR_{0/1}$, $PPIB_{0/1}$, $TPST2_{0/1}$, $CCL4_{0/1}$, $FCGR3A_{0/1}$, $RHOC_{0/1}$, $PTPN22_{0/1}$, $CMC1_{0/1}$, $TRDC_{0/1}$, $CLEC2B_{0/1}$, $SH2D1B_{0/1}$, $B3GNT7_{0/1}$, $TRGC1_{0/1}$, $B2M_{1/3}$, $HLA\text{-}A_{1/3}$, $HLA\text{-}C_{1/3}$, $TPT1_{2/3}$

Cluster 1 (78 atoms) :

$LEF1_{0/1}$, $MAL_{0/1}$, $CCR7_{0/1}$, $TCF7_{0/1}$, $TRABD2A_{0/1}$, $CAMK4_{0/1}$, $RCAN3_{0/1}$, $NOSIP_{0/1}$, $TPT1_{1/3}$, $EEF1B2_{1/3}$, $B2M_{2/3}$, $HLA\text{-}C_{2/3}$, $HLA\text{-}A_{2/3}$, $GNLY_{1/1}$, $KLRB1_{1/1}$, $KLRF1_{1/1}$, $SYTL3_{1/1}$, $IL2RB_{1/1}$, $CTSW_{1/1}$, $GZMB_{1/1}$, $CST7_{1/1}$, $NKG7_{1/1}$, $GZMA_{1/1}$, $CCL5_{1/1}$, $KLRD1_{1/1}$, $GZMM_{1/1}$, $PRF1_{1/1}$, $FGFBP2_{1/1}$, $ABHD17A_{1/1}$, $MATK_{1/1}$, $SPON2_{1/1}$, $EFHD2_{1/1}$, $GZMH_{1/1}$, $APMAP_{1/1}$, $CD99_{1/1}$, $CLIC3_{1/1}$, $PYHIN1_{1/1}$, $CHST12_{1/1}$, $ARL4C_{1/1}$, $DUSP2_{1/1}$, $RUNX3_{1/1}$, $LITAF_{1/1}$, $HOPX_{1/1}$, $CD81_{1/1}$, $C12orf75_{1/1}$, $FLNA_{1/1}$, $CLIC1_{1/1}$, $PLAAT4_{1/1}$, $SYNE2_{1/1}$, $ID2_{1/1}$, $CTSC_{1/1}$, $METRNL_{1/1}$, $ACTN4_{1/1}$, $ITGAL_{1/1}$, $MAP3K8_{1/1}$, $SYNE1_{1/1}$, $CD8A_{1/1}$, $CD8B_{1/1}$, $IL2RG_{1/1}$, $SAMD3_{1/1}$, $APOBEC3G_{1/1}$, $PPIB_{1/1}$, $PPP2R5C_{1/1}$, $LYAR_{1/1}$, $TPST2_{1/1}$, $CCL4_{1/1}$, $FCGR3A_{1/1}$, $RHOC_{1/1}$, $PTPN22_{1/1}$, $CMC1_{1/1}$, $TRDC_{1/1}$, $CLEC2B_{1/1}$,

$SH2D1B_{1/1}$, $B3GNT7_{1/1}$, $TRGC1_{1/1}$, $PFN1_{2/2}$, $ACTG1_{2/2}$, $CFL1_{2/2}$

Cluster 2 (57 atomes) :

$PIK3IP1_{0/1}$, $MGAT4A_{0/1}$, $LEPROTL1_{0/1}$, $HLA\text{-}DRB1_{1/1}$, $HLA\text{-}DPB1_{1/1}$, $HLA\text{-}DPA1_{1/1}$, $HLA\text{-}DRA_{1/1}$, $HLA\text{-}DQB1_{1/1}$, $HLA\text{-}DMB_{1/1}$, $POU2F2_{1/1}$, $HLA\text{-}DMA_{1/1}$, $SNX2_{1/1}$, $MEF2C_{1/1}$, $CD74_{1/1}$, $HLA\text{-}DQA1_{1/1}$, $FCRL1_{1/1}$, $CD79A_{1/1}$, $MS4A1_{1/1}$, $IGHM_{1/1}$, $IGHD_{1/1}$, $TNFRSF13C_{1/1}$, $BANK1_{1/1}$, $CD79B_{1/1}$, $CD22_{1/1}$, $PAX5_{1/1}$, $TCL1A_{1/1}$, $IGKC_{1/1}$, $NIBAN3_{1/1}$, $FCER2_{1/1}$, $TCF4_{1/1}$, $ADAM28_{1/1}$, $RALGPS2_{1/1}$, $BCL11A_{1/1}$, $VPREB3_{1/1}$, $BLK_{1/1}$, $BCL7A_{1/1}$, $COBLL1_{1/1}$, $GNG7_{1/1}$, $HLA\text{-}DOB_{1/1}$, $EBF1_{1/1}$, $AFF3_{1/1}$, $TSPAN13_{1/1}$, $CD40_{1/1}$, $IGLC2_{1/1}$, $PKIG_{1/1}$, $ARHGAP24_{1/1}$, $SPIB_{1/1}$, $BLNK_{1/1}$, $POU2AF1_{1/1}$, $SNX22_{1/1}$, $JCHAIN_{1/1}$, $HVCN1_{1/1}$, $CXCR4_{1/1}$, $IGLC3_{1/1}$, $IL4R_{1/1}$, $CD37_{2/2}$, $BTG1_{2/2}$

Cluster 3 (65 atomes) :

$HLA\text{-}DRB1_{0/1}$, $HLA\text{-}DPB1_{0/1}$, $HLA\text{-}DPA1_{0/1}$, $HLA\text{-}DRA_{0/1}$, $HLA\text{-}DQB1_{0/1}$, $HLA\text{-}DMB_{0/1}$, $POU2F2_{0/1}$, $HLA\text{-}DMA_{0/1}$, $SNX2_{0/1}$, $MEF2C_{0/1}$, $CD74_{0/1}$, $HLA\text{-}DQA1_{0/1}$, $FCRL1_{0/1}$, $CD79A_{0/1}$, $MS4A1_{0/1}$, $IGHM_{0/1}$, $IGHD_{0/1}$, $TNFRSF13C_{0/1}$, $BANK1_{0/1}$, $CD79B_{0/1}$, $CD22_{0/1}$, $PAX5_{0/1}$, $TCL1A_{0/1}$, $IGKC_{0/1}$, $NIBAN3_{0/1}$, $FCER2_{0/1}$, $TCF4_{0/1}$, $ADAM28_{0/1}$, $RALGPS2_{0/1}$, $BCL11A_{0/1}$, $VPREB3_{0/1}$, $BLK_{0/1}$, $BCL7A_{0/1}$, $COBLL1_{0/1}$, $GNG7_{0/1}$, $HLA\text{-}DOB_{0/1}$, $EBF1_{0/1}$, $AFF3_{0/1}$, $TSPAN13_{0/1}$, $CD40_{0/1}$, $IGLC2_{0/1}$, $PKIG_{0/1}$, $ARHGAP24_{0/1}$, $SPIB_{0/1}$, $BLNK_{0/1}$, $POU2AF1_{0/1}$, $SNX22_{0/1}$, $JCHAIN_{0/1}$, $HVCN1_{0/1}$, $CXCR4_{0/1}$, $IGLC3_{0/1}$, $IL4R_{0/1}$ $S100A4_{1/2}$, $CD37_{1/2}$, $LEF1_{1/1}$, $CCR7_{1/1}$, $MAL_{1/1}$, $TCF7_{1/1}$, $TRABD2A_{1/1}$, $PIK3IP1_{1/1}$, $NOSIP_{1/1}$, $CAMK4_{1/1}$, $RCAN3_{1/1}$, $MGAT4A_{1/1}$, $LEPROTL1_{1/1}$

Cluster 4 (268 atomes) :

$FLNA_{0/1}$, $S100A4_{0/2}$, $ZEB2_{0/1}$, $MYO1F_{0/1}$, $AHNAK_{0/1}$, $S100A11_{0/1}$, $S100A9_{0/1}$, $LYZ_{0/1}$, $FTL_{0/1}$, $CTSS_{0/1}$, $S100A8_{0/1}$, $CST3_{0/1}$, $FOS_{0/1}$, $CYBB_{0/1}$, $LST1_{0/1}$, $FCN1_{0/1}$, $S100A12_{0/1}$, $VCAN_{0/1}$, $FGL2_{0/1}$, $PSAP_{0/1}$, $TYMP_{0/1}$, $TYROBP_{0/1}$, $MNDA_{0/1}$, $CSTA_{0/1}$, $COTL1_{0/1}$, $SERPINA1_{0/1}$, $AIF1_{0/1}$, $AP1S2_{0/1}$, $TKT_{0/1}$, $ASAH1_{0/1}$, $APLP2_{0/1}$, $NAMPT_{0/1}$, $MS4A6A_{0/1}$, $FCER1G_{0/1}$, $NCF1_{0/1}$, $LGALS1_{0/1}$, $GRN_{0/1}$, $BRI3_{0/1}$, $DUSP1_{0/1}$, $RNF130_{0/1}$, $CTSD_{0/1}$, $KLF10_{0/1}$, $CSF3R_{0/1}$, $CEBPB_{0/1}$, $CEBPD_{0/1}$, $STXBP2_{0/1}$, $MCL1_{0/1}$, $CLEC7A_{0/1}$, $CD14_{0/1}$, $SPI1_{0/1}$, $TSPO_{0/1}$, $MPEG1_{0/1}$, $CD36_{0/1}$, $LRRK2_{0/1}$, $CD68_{0/1}$, $RGS2_{0/1}$, $CTSB_{0/1}$, $FGR_{0/1}$,

$SNX10_{0/1}$, $NCF2_{0/1}$, $VSIR_{0/1}$, $EVI2B_{0/1}$, $KCTD12_{0/1}$, $EGR1_{0/1}$, $SAT1_{0/1}$, $SOD2_{0/1}$, $IER2_{0/1}$, $DPYSL2_{0/1}$, $CLEC12A_{0/1}$, $FKBP1A_{0/1}$, $CTSZ_{0/1}$, $TNFSF13B_{0/1}$, $GSTP1_{0/1}$, $TNFRSF1B_{0/1}$, $GRINA_{0/1}$, $SKAP2_{0/1}$, $CYP1B1_{0/1}$, $SDCBP_{0/1}$, $MEGF9_{0/1}$, $PGD_{0/1}$, $HIF1A_{0/1}$, $ACTR2_{0/1}$, $LTA4H_{0/1}$, $IRAK3_{0/1}$, $ZFAND5_{0/1}$, $PPT1_{0/1}$, $NPC2_{0/1}$, $CD44_{0/1}$, $RTN3_{0/1}$, $NFKBIA_{0/1}$, $SERPINB1_{0/1}$, $JAML_{0/1}$, $IRS2_{0/1}$, $LYN_{0/1}$, $RAC1_{0/1}$, $FCGRT_{0/1}$, $NCOA4_{0/1}$, $FPR1_{0/1}$, $OGFRL1_{0/1}$, $ZYX_{0/1}$, $DPYD_{0/1}$, $PICALM_{0/1}$, $ZFP36_{0/1}$, $PLBD1_{0/1}$, $RAB31_{0/1}$, $CFD_{0/1}$, $ANXA1_{0/1}$, $LAMP2_{0/1}$, $C1orf162_{0/1}$, $DMXL2_{0/1}$, $ADA2_{0/1}$, $PTPRE_{0/1}$, $CMIP_{0/1}$, $IER5_{0/1}$, $G0S2_{0/1}$, $CD302_{0/1}$, $ANXA5_{0/1}$, $THBS1_{0/1}$, $TPP1_{0/1}$, $PYCARD_{0/1}$, $QKI_{0/1}$, $YBX3_{0/1}$, $H2AFY_{0/1}$, $NUP214_{0/1}$, $ARRB2_{0/1}$, $AGTRAP_{0/1}$, $IQGAP1_{0/1}$, $CASP1_{0/1}$, $NOTCH2_{0/1}$, $VIM_{0/1}$, $GCA_{0/1}$, $ATP6V0D1_{0/1}$, $CHP1_{0/1}$, $MARCKS_{0/1}$, $EHBP1L1_{0/1}$, $LYST_{0/1}$, $CRISPLD2_{0/1}$, $CD4_{0/1}$, $FOSB_{0/1}$, $NINJ1_{0/1}$, $GASK1B_{0/1}$, $ALDH2_{0/1}$, $ANPEP_{0/1}$, $GDI2_{0/1}$, $SCPEP1_{0/1}$, $MGAT1_{0/1}$, $CPVL_{0/1}$, $CFP_{0/1}$, $LGALS3_{0/1}$, $LGALS9_{0/1}$, $RGS18_{0/1}$, $FCAR_{0/1}$, $ANXA2_{0/1}$, $LILRA5_{0/1}$, $CD93_{0/1}$, $MYADM_{0/1}$, $C5AR1_{0/1}$, $SLC11A1_{0/1}$, $ARPC5_{0/1}$, $TGFBI_{0/1}$, $PLXDC2_{0/1}$, $ALDH1A1_{0/1}$, $CRTAP_{0/1}$, $CD163_{0/1}$, $SCO2_{0/1}$, $PLEKHO1_{0/1}$, $TIMP1_{0/1}$, $STX11_{0/1}$, $TCIRG1_{0/1}$, $DUSP6_{0/1}$, $TMEM176B_{0/1}$, $NUMB_{0/1}$, $SGK1_{0/1}$, $TREM1_{0/1}$, $MARCH1_{0/1}$, $RILPL2_{0/1}$, $SH3BP2_{0/1}$, $ARPC1B_{0/1}$, $AOAH_{0/1}$, $BLVRB_{0/1}$, $PECAM1_{0/1}$, $GLUL_{0/1}$, $LAP3_{0/1}$, $WARS_{0/1}$, $TNFAIP2_{0/1}$, $MIDN_{0/1}$, $TFEC_{0/1}$, $THEMIS2_{0/1}$, $IFNGR1_{0/1}$, $CAPG_{0/1}$, $HSBP1_{0/1}$, $CUX1_{0/1}$, $SEC14L1_{0/1}$, $SLC7A7_{0/1}$, $KLF4_{0/1}$, $SIRPA_{0/1}$, $SH3BGRL_{0/1}$, $PRELID1_{0/1}$, $SAMHD1_{0/1}$, $FOSL2_{0/1}$, $GABARAP_{0/1}$, $IFNGR2_{0/1}$, $F13A1_{0/1}$, $TLR4_{0/1}$, $RIN3_{0/1}$, $IFITM3_{0/1}$, $SRGN_{0/1}$, $JDP2_{0/1}$, $HLA\text{-}DRB5_{0/1}$, $APOBEC3A_{0/1}$, $FCGR1A_{0/1}$, $CXCL16_{0/1}$, $CAP1_{0/1}$, $LGALS2_{0/1}$, $CSF2RA_{0/1}$, $MS4A7_{0/1}$, $FCGR2A_{0/1}$, $NR4A1_{0/1}$, $LILRB2_{0/1}$, $TNFSF10_{0/1}$, $TLR8_{0/1}$, $BID_{0/1}$, $CSF1R_{0/1}$, $CDKN1C_{0/1}$, $SIGLEC10_{0/1}$, $PILRA_{0/1}$, $ZNF703_{0/1}$, $LRRC25_{0/1}$, $PLSCR1_{0/1}$, $GAS7_{0/1}$, $CD300E_{0/1}$, $CCDC88A_{0/1}$, $GPBAR1_{0/1}$, $CPPED1_{0/1}$, $HCK_{0/1}$, $CAMK1_{0/1}$, $CTSH_{0/1}$, $PGK1_{0/1}$, $ALDH3B1_{0/1}$, $MYOF_{0/1}$, $TCF7L2_{0/1}$, $HK3_{0/1}$, $MAFB_{0/1}$, $TMEM176A_{0/1}$, $HBEGF_{0/1}$, $IFI30_{0/1}$, $PIK3AP1_{0/1}$, $HMOX1_{0/1}$, $CTSL_{0/1}$, $LMO2_{0/1}$, $C19orf38_{0/1}$, $LILRB1_{0/1}$, $RRAS_{0/1}$, $ACTB_{1/3}$, $LTB_{1/2}$, $CD3E_{1/2}$, $S100A6_{1/2}$, $IL32_{1/2}$, $EEF1B2_{2/3}$, $IL7R_{1/1}$, $C12orf57_{1/1}$, $TRAC_{1/1}$, $CD2_{1/1}$, $LAT_{1/1}$, $CD6_{1/1}$, $PRKCH_{1/1}$, $BCL2_{1/1}$, $CD27_{1/1}$

Cluster 5 (267 atomes) :

$CD3E_{0/2}$, $IL32_{0/2}$, $TLE5_{0/2}$, $LTB_{0/2}$, $IL7R_{0/1}$, $TRAC_{0/1}$, $C12orf57_{0/1}$, $CD2_{0/1}$, $LAT_{0/1}$, $CD6_{0/1}$, $PRKCH_{0/1}$, $BCL2_{0/1}$, $CD27_{0/1}$, $ACTB_{2/3}$, $ZEB2_{1/1}$, $TYROBP_{1/1}$, $S100A9_{1/1}$, $LYZ_{1/1}$, $FTL_{1/1}$, $CTSS_{1/1}$, $S100A8_{1/1}$, $CST3_{1/1}$, $FOS_{1/1}$, $CYBB_{1/1}$, $LST1_{1/1}$, $FCN1_{1/1}$,

$S100A12_{1/1}$, $VCAN_{1/1}$, $FGL2_{1/1}$, $PSAP_{1/1}$, $TYMP_{1/1}$, $MNDA_{1/1}$, $CSTA_{1/1}$, $COTL1_{1/1}$, $SERPINA1_{1/1}$, $AIF1_{1/1}$, $AP1S2_{1/1}$, $TKT_{1/1}$, $ASAH1_{1/1}$, $APLP2_{1/1}$, $NAMPT_{1/1}$, $MS4A6A_{1/1}$, $FCER1G_{1/1}$, $NCF1_{1/1}$, $LGALS1_{1/1}$, $GRN_{1/1}$, $BRI3_{1/1}$, $DUSP1_{1/1}$, $RNF130_{1/1}$, $CTSD_{1/1}$, $KLF10_{1/1}$, $CSF3R_{1/1}$, $CEBPB_{1/1}$, $CEBPD_{1/1}$, $STXBP2_{1/1}$, $MCL1_{1/1}$, $CLEC7A_{1/1}$, $CD14_{1/1}$, $SPI1_{1/1}$, $TSPO_{1/1}$, $MPEG1_{1/1}$, $CD36_{1/1}$, $LRRK2_{1/1}$, $CD68_{1/1}$, $RGS2_{1/1}$, $CTSB_{1/1}$, $S100A11_{1/1}$, $FGR_{1/1}$, $SNX10_{1/1}$, $NCF2_{1/1}$, $VSIR_{1/1}$, $EVI2B_{1/1}$, $KCTD12_{1/1}$, $EGR1_{1/1}$, $SAT1_{1/1}$, $SOD2_{1/1}$, $IER2_{1/1}$, $DPYSL2_{1/1}$, $CLEC12A_{1/1}$, $FKBP1A_{1/1}$, $CTSZ_{1/1}$, $TNFSF13B_{1/1}$, $GSTP1_{1/1}$, $TNFRSF1B_{1/1}$, $GRINA_{1/1}$, $SKAP2_{1/1}$, $CYP1B1_{1/1}$, $SDCBP_{1/1}$, $MEGF9_{1/1}$, $PGD_{1/1}$, $HIF1A_{1/1}$, $ACTR2_{1/1}$, $LTA4H_{1/1}$, $IRAK3_{1/1}$, $ZFAND5_{1/1}$, $PPT1_{1/1}$, $NPC2_{1/1}$, $CD44_{1/1}$, $RTN3_{1/1}$, $NFKBIA_{1/1}$, $SERPINB1_{1/1}$, $JAML_{1/1}$, $IRS2_{1/1}$, $LYN_{1/1}$, $RAC1_{1/1}$, $FCGRT_{1/1}$, $NCOA4_{1/1}$, $FPR1_{1/1}$, $OGFRL1_{1/1}$, $ZYX_{1/1}$, $DPYD_{1/1}$, $PICALM_{1/1}$, $ZFP36_{1/1}$, $PLBD1_{1/1}$, $RAB31_{1/1}$, $CFD_{1/1}$, $ANXA1_{1/1}$, $LAMP2_{1/1}$, $C1orf162_{1/1}$, $DMXL2_{1/1}$, $ADA2_{1/1}$, $PTPRE_{1/1}$, $CMIP_{1/1}$, $IER5_{1/1}$, $G0S2_{1/1}$, $CD302_{1/1}$, $ANXA5_{1/1}$, $THBS1_{1/1}$, $TPP1_{1/1}$, $PYCARD_{1/1}$, $QKI_{1/1}$, $YBX3_{1/1}$, $H2AFY_{1/1}$, $NUP214_{1/1}$, $ARRB2_{1/1}$, $AGTRAP_{1/1}$, $IQGAP1_{1/1}$, $CASP1_{1/1}$, $NOTCH2_{1/1}$, $VIM_{1/1}$, $GCA_{1/1}$, $ATP6V0D1_{1/1}$, $CHP1_{1/1}$, $MARCKS_{1/1}$, $EHBP1L1_{1/1}$, $LYST_{1/1}$, $CRISPLD2_{1/1}$, $AHNAK_{1/1}$, $CD4_{1/1}$, $FOSB_{1/1}$, $NINJ1_{1/1}$, $GASK1B_{1/1}$, $ALDH2_{1/1}$, $ANPEP_{1/1}$, $GDI2_{1/1}$, $SCPEP1_{1/1}$, $MGAT1_{1/1}$, $CPVL_{1/1}$, $CFP_{1/1}$, $LGALS3_{1/1}$, $LGALS9_{1/1}$, $RGS18_{1/1}$, $FCAR_{1/1}$, $ANXA2_{1/1}$, $LILRA5_{1/1}$, $CD93_{1/1}$, $MYADM_{1/1}$, $C5AR1_{1/1}$, $SLC11A1_{1/1}$, $ARPC5_{1/1}$, $TGFBI_{1/1}$, $PLXDC2_{1/1}$, $ALDH1A1_{1/1}$, $CRTAP_{1/1}$, $CD163_{1/1}$, $SCO2_{1/1}$, $PLEKHO1_{1/1}$, $TIMP1_{1/1}$, $STX11_{1/1}$, $TCIRG1_{1/1}$, $DUSP6_{1/1}$, $TMEM176B_{1/1}$, $NUMB_{1/1}$, $MYO1F_{1/1}$, $SGK1_{1/1}$, $TREM1_{1/1}$, $MARCH1_{1/1}$, $RILPL2_{1/1}$, $SH3BP2_{1/1}$, $AOAH_{1/1}$, $BLVRB_{1/1}$, $PECAM1_{1/1}$, $GLUL_{1/1}$, $LAP3_{1/1}$, $WARS_{1/1}$, $TNFAIP2_{1/1}$, $MIDN_{1/1}$, $TFEC_{1/1}$, $THEMIS2_{1/1}$, $IFNGR1_{1/1}$, $CAPG_{1/1}$, $HSBP1_{1/1}$, $CUX1_{1/1}$, $SEC14L1_{1/1}$, $SLC7A7_{1/1}$, $KLF4_{1/1}$, $SIRPA_{1/1}$, $SH3BGRL_{1/1}$, $PRELID1_{1/1}$, $SAMHD1_{1/1}$, $FOSL2_{1/1}$, $GABARAP_{1/1}$, $IFNGR2_{1/1}$, $F13A1_{1/1}$, $TLR4_{1/1}$, $RIN3_{1/1}$, $IFITM3_{1/1}$, $SRGN_{1/1}$, $JDP2_{1/1}$, $HLA\text{-}DRB5_{1/1}$, $APOBEC3A_{1/1}$, $FCGR1A_{1/1}$, $CXCL16_{1/1}$, $CAP1_{1/1}$, $LGALS2_{1/1}$, $CSF2RA_{1/1}$, $MS4A7_{1/1}$, $FCGR2A_{1/1}$, $NR4A1_{1/1}$, $LILRB2_{1/1}$, $TNFSF10_{1/1}$, $TLR8_{1/1}$, $BID_{1/1}$, $CSF1R_{1/1}$, $CDKN1C_{1/1}$, $SIGLEC10_{1/1}$, $PILRA_{1/1}$, $ZNF703_{1/1}$, $LRRC25_{1/1}$, $PLSCR1_{1/1}$, $GAS7_{1/1}$, $CD300E_{1/1}$, $CCDC88A_{1/1}$, $GPBAR1_{1/1}$, $CPPED1_{1/1}$, $HCK_{1/1}$, $CAMK1_{1/1}$, $CTSH_{1/1}$, $PGK1_{1/1}$, $ALDH3B1_{1/1}$, $MYOF_{1/1}$, $TCF7L2_{1/1}$, $HK3_{1/1}$, $MAFB_{1/1}$, $TMEM176A_{1/1}$, $HBEGF_{1/1}$, $IFI30_{1/1}$, $PIK3AP1_{1/1}$, $HMOX1_{1/1}$, $CTSL_{1/1}$, $LMO2_{1/1}$, $C19orf38_{1/1}$, $LILRB1_{1/1}$, $RRAS_{1/1}$, $S100A6_{2/2}$, $FTH1_{2/2}$, $S100A4_{2/2}$

F.2 Clusters sur le graphe du sous-jeu de données (myéloïdes)

Cluster 2 (57 atomes) :

FOS$_{0/1}$, *VCAN*$_{0/1}$, *S100A12*$_{0/1}$, *S100A8*$_{0/1}$, *MS4A6A*$_{0/1}$, *LYZ*$_{0/1}$, *CSF3R*$_{0/1}$, *CD14*$_{0/1}$, *SLC2A3*$_{0/1}$, *CD36*$_{0/1}$, *EGR1*$_{0/1}$, *IER2*$_{0/1}$, *S100A9*$_{0/1}$, *KLF10*$_{0/1}$, *MNDA*$_{0/1}$, *CD163*$_{0/1}$, *MEGF9*$_{0/1}$, *HES4*$_{1/1}$, *FCGR3A*$_{1/1}$, *CDKN1C*$_{1/1}$, *RHOC*$_{1/1}$, *SIGLEC10*$_{1/1}$, *TCF7L2*$_{1/1}$, *CSF1R*$_{1/1}$, *IFITM2*$_{1/1}$, *MS4A7*$_{1/1}$, *CTSL*$_{1/1}$, *PILRA*$_{1/1}$, *LRRC25*$_{1/1}$, *IFITM3*$_{1/1}$, *PAG1*$_{1/1}$, *DRAP1*$_{1/1}$, *LILRB2*$_{1/1}$, *SPN*$_{1/1}$, *HMOX1*$_{1/1}$, *PTP4A3*$_{1/1}$, *ZNF703*$_{1/1}$, *LY6E*$_{1/1}$, *RRAS*$_{1/1}$, *BATF3*$_{1/1}$, *VMO1*$_{1/1}$, *PPM1N*$_{1/1}$, *SLC44A2*$_{1/1}$, *C1QA*$_{1/1}$, *CKB*$_{1/1}$, *WARS*$_{1/1}$, *OAS1*$_{1/1}$, *NAP1L1*$_{1/1}$, *MS4A4A*$_{1/1}$, *SYTL1*$_{1/1}$, *ISG15*$_{1/1}$, *UNC119*$_{1/1}$, *IL3RA*$_{1/1}$, *IFI6*$_{1/1}$, *ICAM4*$_{1/1}$, *S100A4*$_{2/2}$, *S100A6*$_{2/2}$

Cluster 3 (18 atomes) :

NCF1$_{0/1}$, *GBP2*$_{0/1}$, *APOBEC3A*$_{0/1}$, *CTSS*$_{0/1}$, *FTL*$_{0/1}$, *PSAP*$_{0/1}$, *CYBB*$_{0/1}$, *HLA-DQA1*$_{1/1}$, *CD74*$_{1/1}$, *HLA-DRB5*$_{1/1}$, *FCER1A*$_{1/1}$, *NDRG2*$_{1/1}$, *CD1C*$_{1/1}$, *ENHO*$_{1/1}$, *CLIC2*$_{1/1}$, *AREG*$_{1/1}$, *HLA-DPB1*$_{1/1}$, *HLA-DPA1*$_{1/1}$

Cluster 5 (13 atomes) :

PF4$_{1/1}$, *CAVIN2*$_{1/1}$, *PPBP*$_{1/1}$, *TUBB1*$_{1/1}$, *GNG11*$_{1/1}$, *NRGN*$_{1/1}$, *HIST1H2AC*$_{1/1}$, *SPARC*$_{1/1}$, *TMEM40*$_{1/1}$, *CLU*$_{1/1}$, *GP9*$_{1/1}$, *PTCRA*$_{1/1}$, *MPIG6B*$_{1/1}$

Cluster 6 (18 atomes) :

CD3E$_{1/2}$, *LTB*$_{1/2}$, *IL32*$_{1/2}$, *IL7R*$_{1/1}$, *SLC38A1*$_{1/1}$, *PRKCH*$_{1/1}$, *SKAP1*$_{1/1}$, *IKZF3*$_{1/1}$, *LEF1*$_{1/1}$, *STAT4*$_{1/1}$, *CST7*$_{1/1}$, *GZMA*$_{1/1}$, *CCL5*$_{1/1}$, *NKG7*$_{1/1}$, *TRAC*$_{1/1}$, *LYAR*$_{1/1}$, *RHOH*$_{1/1}$, *SYNE1*$_{1/1}$

Annexe G

Recherche de paramètres pour l'extraction de transitions

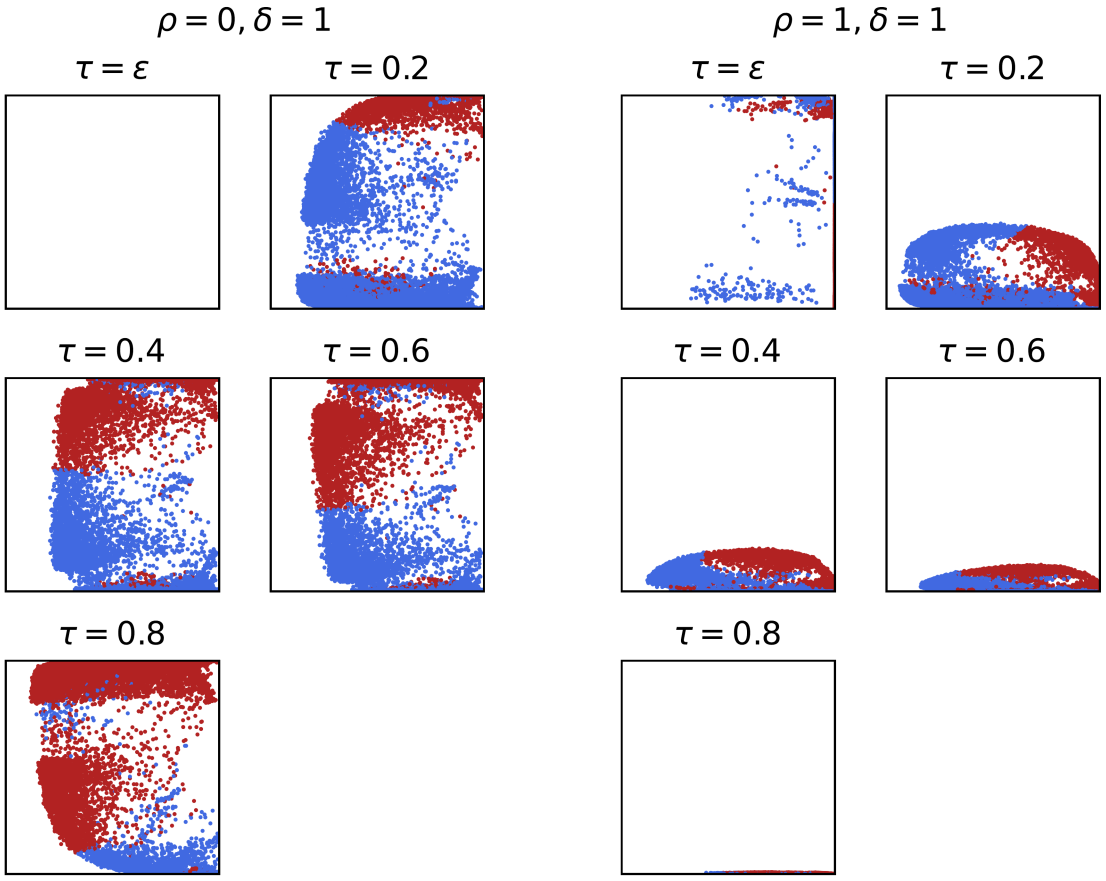

Figure G.1 – Résultat du calcul de qaulité des instances *LOLH* formées à partir des transitions pour les différentes valeurs de τ, avec $\delta = 1$ et $\rho = 0$ à gauche et $\rho = 1$ à droite. Les points rouges correspondent aux atomes avec la valeur discrète 0 et les points bleus correspondent aux atomes ayant une valeur discrète strictement positive.

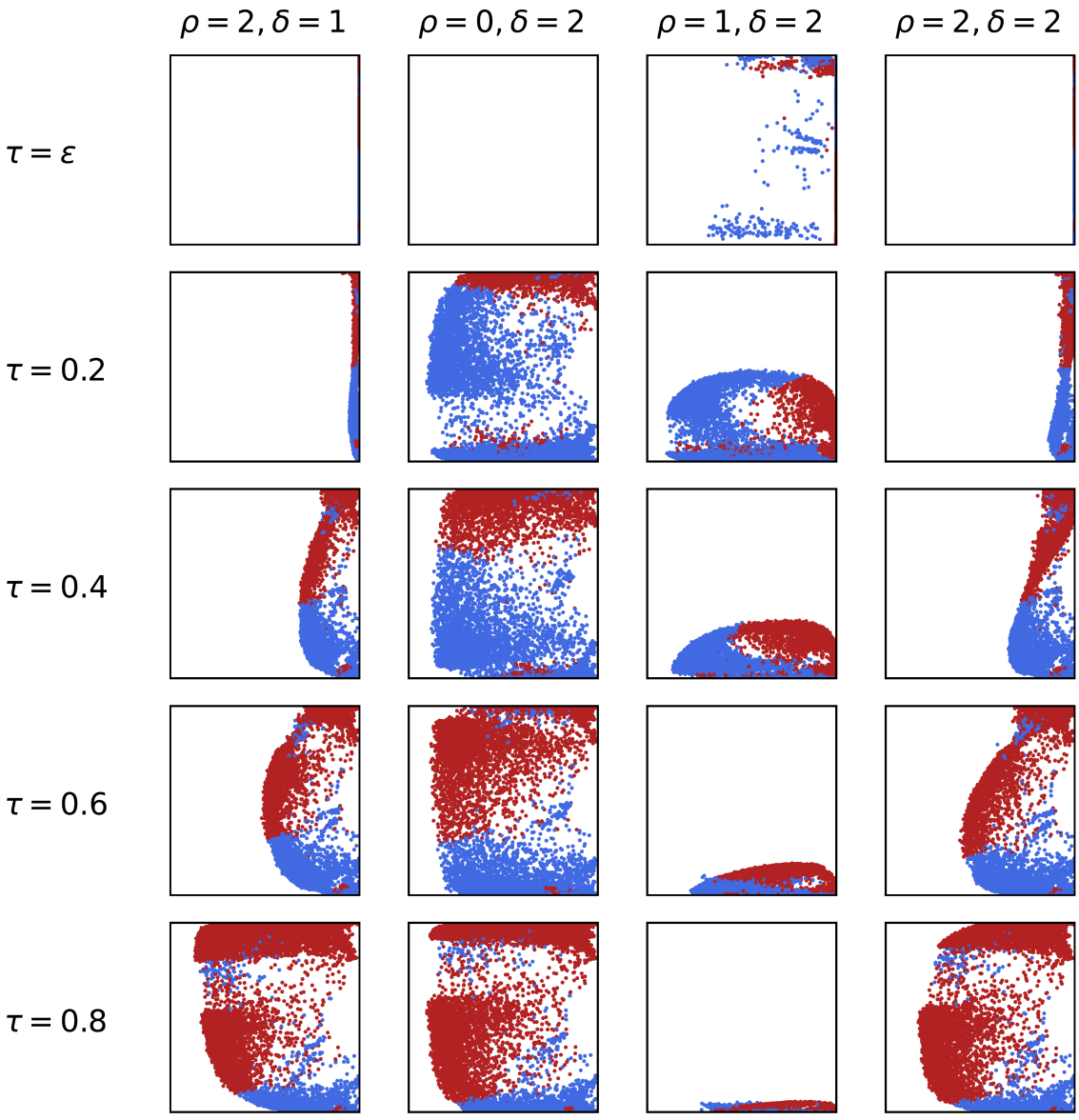

FIGURE G.2 – Résultat du calcul de qualité des instances *LOLH* formées à partir des transitions pour les différentes valeurs de τ, avec de gauche à droite les valeurs suivantes pour ρ et δ : $(\rho = 2, \delta = 1)$, $(\rho = 0, \delta = 2)$, $(\rho = 1, \delta = 2)$, et $(\rho = 2, \delta = 2)$.

BIBLIOGRAPHIE

Akutsu, T., Miyano, S., & Kuhara, S., (2000), Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics*, *16 8*, 727-734, https://doi.org/10.1093/bioinformatics/16.8.727

Akutsu, T., Kosub, S., Melkman, A. A., & Tamura, T., (2012), Finding a Periodic Attractor of a Boolean Network, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9 5*, 1410-1421, https://doi.org/10.1109/TCBB.2012.87

Allen, M., (2015), Compelled by the diagram : thinking through CH Waddington's epigenetic landscape, *Contemporaneity*, *4*, 119.

Aneja, Y. P., & Nair, K. P. K., (1979), Bicriteria Transportation Problem, *Management Science*, *25 1*, 73-78, https://doi.org/10.1287/mnsc.25.1.73

Assmann, S. M., & Albert, R., (2009), Discrete Dynamic Modeling with Asynchronous Update, or How to Model Complex Systems in the Absence of Quantitative Information, *In* D. A. Belostotsky (Éd.), *Plant Systems Biology* (p. 207-225), Humana Press, https://doi.org/10.1007/978-1-60327-563-7_10

Aubin-Frankowski, P.-C., & Vert, J.-P., (2020), Gene regulation inference from single-cell RNA-seq data with linear differential equations and velocity inference (J. Gorodkin, Éd.), *Bioinformatics*, *36 18*, 4774-4780, https://doi.org/10.1093/bioinformatics/btaa576

Audemard, G., & Simon, L., (2014), Lazy Clause Exchange Policy for Parallel SAT Solvers, *In* D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, C. Sinz & U. Egly (Éd.), *Theory and Applications of Satisfiability Testing – SAT 2014* (p. 197-205), Springer International Publishing, https://doi.org/10.1007/978-3-319-09284-3_15

Barbosa, S., Niebel, B., Wolf, S., Mauch, K., & Takors, R., (2018), A guide to gene regulatory network inference for obtaining predictive solutions : Underlying assumptions and fundamental biological and data constraints, *Biosystems*, *174*, 37-48, https://doi.org/10.1016/j.biosystems.2018.10.008

Barman, S., & Kwon, Y.-K., (2018), A Boolean network inference from time-series gene expression data using a genetic algorithm, *Bioinformatics*, *34 17*, i927-i933, https://doi.org/10.1093/bioinformatics/bty584

Behaegel, J., Comet, J.-P., Bernot, G., Cornillon, E., & Delaunay, F., (2016), A hybrid model of cell cycle in mammals, *Journal of Bioinformatics and Computational Biology*, *14 01*, 1640001, https://doi.org/10.1142/S0219720016400011

Ben Abdallah, E., Folschette, M., Roux, O., & Magnin, M., (2015), Exhaustive analysis of dynamical properties of Biological Regulatory Networks with Answer Set Programming, *2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 281-285, https://doi.org/10.1109/BIBM.2015.7359694

Ben Abdallah, E., Ribeiro, T., Magnin, M., Roux, O., & Inoue, K., (2017a), Modeling Delayed Dynamics in Biological Regulatory Networks from Time Series Data, *Algorithms*, *10 1*, 8, https://doi.org/10.3390/a10010008

Ben Abdallah, E., Folschette, M., Roux, O., & Magnin, M., (2017b), ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks, *Algorithms for Molecular Biology*, *12 1*, 20, https://doi.org/10.1186/s13015-017-0111-2

Bergen, V., Lange, M., Peidli, S., Wolf, F. A., & Theis, F. J., (2020), Generalizing RNA velocity to transient cell states through dynamical modeling, *Nature Biotechnology*, *38 12*, 1408-1414, https://doi.org/10.1038/s41587-020-0591-3

Bergen, V., Soldatov, R. A., Kharchenko, P. V., & Theis, F. J., (2021), RNA velocity—current challenges and future perspectives, *Molecular Systems Biology*, *17 8*, https://doi.org/10.15252/msb.202110282

Bich, L., Mossio, M., Ruiz-Mirazo, K., & Moreno, A., (2016), Biological regulation : controlling the system from within, *Biology & Philosophy*, *31 2*, 237-265, https://doi.org/10.1007/s10539-015-9497-8

Biere, A., Heule, M., van Maaren, H., & Walsh, T., (2009), *Handbook of Satisfiability : Volume 185 Frontiers in Artificial Intelligence and Applications*, IOS Press.

Blasi, M. F., Casorelli, I., Colosimo, A., Blasi, F. S., Bignami, M., & Giuliani, A., (2005), A recursive network approach can identify constitutive regulatory circuits in gene expression data, *Physica A : Statistical Mechanics and its Applications*, *348*, 349-370, https://doi.org/10.1016/j.physa.2004.09.005

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E., (2008), Fast unfolding of communities in large networks, *Journal of Statistical Mechanics : Theory and Experiment*, *2008 10*, P10008, https://doi.org/10.1088/1742-5468/2008/10/P10008

Bornholdt, S., (2008), Boolean network models of cellular regulation : prospects and limitations, *Journal of The Royal Society Interface*, *5 suppl_1*, https://doi.org/10.1098/rsif.2008.0132.focus

Bryant, R. E., (1986), Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers*, *C-35 8*, 677-691, https://doi.org/10.1109/TC.1986.1676819

Buchet, S., Carbone, F., Magnin, M., Ménager, M., & Roux, O., (2021), Inference of Gene Networks from Single Cell Data through Quantified Inductive Logic Programming, *The 12th International Conference on Computational Systems-Biology and Bioinformatics*, 48-63, https://doi.org/10.1145/3486713.3486746

Burch, J. R., Clarke, E. M., McMillan, K. L., Dill, D. L., & Hwang, L. J., (1990), Symbolic model checking : 10^{20} states and beyond, *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science (LICS 1990)*, 428-439.

Camin, J. H., & Sokal, R. R., (1965), A Method for Deducing Branching Sequences in Phylogeny, *Evolution*, *19 3*, 311-326, http://www.jstor.org/stable/2406441

Campbell, C., & Albert, R., (2014), Stabilization of perturbed Boolean network attractors through compensatory interactions, *BMC Systems Biology*, *8 1*, 53, https://doi.org/10.1186/1752-0509-8-53

Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D. M., Hill, A. J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F. J., Trapnell, C., & Shendure, J., (2019), The single-cell transcriptional landscape of mammalian organogenesis, *Nature*, *566 7745*, 496-502, https://doi.org/10.1038/s41586-019-0969-x

Češka, M., Šafránek, D., Dražan, S., & Brim, L., (2014), Robustness Analysis of Stochastic Biochemical Systems, *PLOS ONE*, *9 4*, 1-23, https://doi.org/10.1371/journal.pone.0094553

Chaouiya, C., Naldi, A., & Thieffry, D., (2012), Logical Modelling of Gene Regulatory Networks with GINsim, *In* J. van Helden, A. Toussaint & D. Thieffry (Éd.), *Bacterial Molecular Networks : Methods and Protocols* (p. 463-479), Springer New York, https://doi.org/10.1007/978-1-61779-361-5_23

Chatain, T., Haar, S., Jezequel, L., Paulevé, L., & Schwoon, S., (2014), Characterization of Reachable Attractors Using Petri Net Unfoldings, *In* P. Mendes, J. O. Dada & K. Smallbone (Éd.), *Computational Methods in Systems Biology* (p. 129-142), Springer International Publishing.

Chen, H., Guo, J., Mishra, S. K., Robson, P., Niranjan, M., & Zheng, J., (2015), Single-cell transcriptional analysis to uncover regulatory circuits driving cell fate decisions in early mouse development, *Bioinformatics*, *31 7*, 1060-1066, https://doi.org/10.1093/bioinformatics/btu777

Chen, L., Qi-Wei, G., Nakata, M., Matsuno, H., & Miyano, S., (2007), Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets, *Journal of Biosciences*, *32 1*, 113-127, https://doi.org/10.1007/s12038-007-0011-6

Chen, S., & Mar, J. C., (2018), Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data, *BMC Bioinformatics*, *19 1*, 232, https://doi.org/10.1186/s12859-018-2217-z

Cheng, A., Esparza, J., & Palsberg, J., (1995), Complexity results for 1-safe nets, *Theoretical Computer Science*, *1471*, 117-136, https://doi.org/https://doi.org/10.1016/0304-3975(94)00231-7

Chevalier, S., Froidevaux, C., Pauleve, L., & Zinovyev, A., (2019), Synthesis of Boolean Networks from Biological Dynamical Constraints using Answer-Set Programming, *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 34-41, https://doi.org/10.1109/ICTAI.2019.00014

Cimatti, A., Clarke, E., Giunchiglia, F., & Roveri, M., (1999), NuSMV : A New Symbolic Model Verifier, *Computer Aided Verification*, 495-499, https://doi.org/10.1007/3-540-48683-6_44

Clarke, E. M., (2008), The Birth of Model Checking, *In* O. Grumberg & H. Veith (Éd.), *25 Years of Model Checking : History, Achievements, Perspectives* (p. 1-26), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-540-69850-0_1

Comet, J.-P., Klaudel, H., & Liauzu, S., (2005), Modeling Multi-valued Genetic Regulatory Networks Using High-Level Petri Nets, *In* G. Ciardo & P. Darondeau (Éd.), *Applications and Theory of Petri Nets 2005* (p. 208-227), Springer Berlin Heidelberg.

Comet, J.-P., Fromentin, J., Bernot, G., & Roux, O., (2010), A Formal Model for Gene Regulatory Networks with Time Delays, *In* J. H. Chan, Y.-S. Ong & S.-B. Cho (Éd.), *Computational Systems-Biology and Bioinformatics* (p. 1-13), Springer Berlin Heidelberg.

Comet, J.-P., Noual, M., Richard, A., Aracena, J., Calzone, L., Demongeot, J., Kaufman, M., Naldi, A., Snoussi, E. H., & Thieffry, D., (2013), On Circuit Functionality in Boolean Networks, *Bulletin of Mathematical Biology*, *756*, 906-919, https://doi.org/10.1007/s11538-013-9829-2

Corblin, F., Fanchon, E., & Trilling, L., (2010), Applications of a formal approach to decipher discrete genetic networks, *BMC Bioinformatics*, *111*, 385, https://doi.org/10.1186/1471-2105-11-385

Corblin, F., Fanchon, E., Trilling, L., Chaouiya, C., & Thieffry, D., (2012), Automatic Inference of Regulatory and Dynamical Properties from Incomplete Gene Interaction and Expression Data, *In* M. A. Lones, S. L. Smith, S. Teichmann, F. Naef, J. A. Walker & M. A. Trefzer (Éd.), *Information Processign in Cells and Tissues* (p. 25-30), Springer Berlin Heidelberg.

Cousot, P., & Cousot, R., (1977), Abstract interpretation : a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '77*, 238-252, https://doi.org/10.1145/512950.512973

Cropper, A., Dumančić, S., & Muggleton, S. H., (2020), Turning 30 : New Ideas in Inductive Logic Programming [Survey track], *In* C. Bessiere (Éd.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20* (p. 4833-4839), International Joint Conferences on Artificial Intelligence Organization, https://doi.org/10.24963/ijcai.2020/673

Danos, V., Feret, J., Fontana, W., Harmer, R., & Krivine, J., (2007), Rule-Based Modelling of Cellular Signalling, *In* L. Caires & V. T. Vasconcelos (Éd.), *CONCUR 2007 – Concurrency Theory* (p. 17-41), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-540-74407-8_3

de Jong, H., (2002), Modeling and Simulation of Genetic Regulatory Systems : A Literature Review, *Journal of Computational Biology*, *9 1*, 67-103, https://doi.org/10.1089/10665270252833208

Delgado, F. M., & Gómez-Vela, F., (2019), Computational methods for Gene Regulatory Networks reconstruction and analysis : A review, *Artificial Intelligence in Medicine*, *95*, 133-145, https://doi.org/10.1016/j.artmed.2018.10.006

Dubrova, E., & Teslenko, M., (2011), A SAT-Based Algorithm for Finding Attractors in Synchronous Boolean Networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *8 5*, 1393-1399, https://doi.org/10.1109/TCBB.2010.20

Eén, N., & Sörensson, N., (2004), An Extensible SAT-solver, *In* E. Giunchiglia & A. Tacchella (Éd.), *Theory and Applications of Satisfiability Testing* (p. 502-518), Springer Berlin Heidelberg.

Elmesmari, A., Fraser, A. R., Wood, C., Gilchrist, D., Vaughan, D., Stewart, L., McSharry, C., McInnes, I. B., & Kurowska-Stolarska, M., (2016), MicroRNA-155 regulates monocyte chemokine and chemokine receptor expression in Rheumatoid Arthritis, *Rheumatology (Oxford)*, *55 11*, 2056-2065.

Fayruzov, T., Janssen, J., Vermeir, D., Cornelis, C., & Cock, M. D., (2011), Modelling gene and protein regulatory networks with Answer Set Programming, *International Journal of Data Mining and Bioinformatics*, *5 2*, 209, https://doi.org/10.1504/IJDMB.2011.039178

Fisher, J., Köksal, A. S., Piterman, N., & Woodhouse, S., (2015), Synthesising Executable Gene Regulatory Networks from Single-Cell Gene Expression Data, *In* D. Kroening & C. S. Păsăreanu (Éd.), *Computer Aided Verification* (p. 544-560), Springer International Publishing.

Folschette, M., Paulevé, L., Inoue, K., Magnin, M., & Roux, O., (2015a), Identification of biological regulatory networks from Process Hitting models, *Theoretical Computer Science*, *568*, 49-71, https://doi.org/https://doi.org/10.1016/j.tcs.2014.12.002

Folschette, M., Paulevé, L., Magnin, M., & Roux, O., (2015b), Sufficient conditions for reachability in automata networks with priorities, *Theoretical Computer Science*, *608*, 66-83, https://doi.org/10.1016/j.tcs.2015.08.040

Fuller, T., Langfelder, P., Presson, A., & Horvath, S., (2011), Review of Weighted Gene Coexpression Network Analysis, *In* H. H.-S. Lu, B. Schölkopf & H. Zhao (Éd.), *Handbook of Statistical Bioinformatics* (p. 369-388), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-16345-6_18

Gallet, E., Manceny, M., Le Gall, P., & Ballarini, P., (2014), An LTL Model Checking Approach for Biological Parameter Inference, *In* S. Merz & J. Pang

(Éd.), *Formal Methods and Software Engineering* (p. 155-170), Springer International Publishing, https://doi.org/10.1007/978-3-319-11737-9_11

Gao, S., Sun, C., Xiang, C., Qin, K., & Lee, T. H., (2020), Learning Asynchronous Boolean Networks From Single-Cell Data Using Multiobjective Cooperative Genetic Programming, *IEEE Transactions on Cybernetics*, 1-15, https://doi.org/10.1109/TCYB.2020.3022430

García-Campos, M. A., Espinal-Enríquez, J., & Hernández-Lemus, E., (2015), Pathway Analysis : State of the Art, *Frontiers in Physiology*, *6*, https://doi.org/10.3389/fphys.2015.00383

Giacobbe, M., Guet, C. C., Gupta, A., Henzinger, T. A., Paixão, T., & Petrov, T., (2017), Model checking the evolution of gene regulatory networks, *Acta Informatica*, *54 8*, 765-787.

Gonzalez, A., Chaouiya, C., & Thieffry, D., (2008), Logical modelling of the role of the Hh pathway in the patterning of the Drosophila wing disc, *Bioinformatics*, *24 16*, i234-i240, https://doi.org/10.1093/bioinformatics/btn266

Hafemeister, C., & Satija, R., (2019), Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression, *Genome Biology*, *20 1*, https://doi.org/10.1186/s13059-019-1874-1

Hamey, F. K., Nestorowa, S., Kinston, S. J., Kent, D. G., Wilson, N. K., & Göttgens, B., (2017), Reconstructing blood stem cell regulatory network models from single-cell molecular profiles, *Proceedings of the National Academy of Sciences*, *114 23*, 5822-5829, https://doi.org/10.1073/pnas.1610609114

Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W. M., Zheng, S., Butler, A., Lee, M. J., Wilk, A. J., Darby, C., Zager, M., Hoffman, P., Stoeckius, M., Papalexi, E., Mimitou, E. P., Jain, J., Srivastava, A., Stuart, T., Fleming, L. M., Yeung, B., ... Satija, R., (2021), Integrated analysis of multimodal single-cell data, *Cell*, *184 13*, 3573-3587.e29, https://doi.org/https://doi.org/10.1016/j.cell.2021.04.048

Heath, J., Kwiatkowska, M., Norman, G., Parker, D., & Tymchyshyn, O., (2008), Probabilistic model checking of complex biological pathways, *Theoretical Computer Science*, *391 3*, 239-257.

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E., & Guthke, R., (2009), Gene regulatory network inference : Data integration in dynamic models—A review, *Biosystems*, *96 1*, 86-103, https://doi.org/10.1016/j.biosystems.2008.12.004

Inoue, K., Ribeiro, T., & Sakama, C., (2014), Learning from interpretation transition, *Machine Learning*, *94 1*, 51-79, https://doi.org/10.1007/s10994-013-5353-8

Jacob, F., & Monod, J., (1961), Genetic regulatory mechanisms in the synthesis of proteins, *Journal of Molecular Biology*, *3 3*, 318-356, https://doi.org/10.1016/S0022-2836(61)80072-7

Jacomy, M., Venturini, T., Heymann, S., & Bastian, M., (2014), ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software, *PLOS ONE*, *9 6*, 1-12, https://doi.org/10.1371/journal.pone.0098679

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., ... Hassabis, D., (2021), Highly accurate protein structure prediction with AlphaFold, *Nature*, https://doi.org/10.1038/s41586-021-03819-2

Kang, Y., Thieffry, D., & Cantini, L., (2021), Evaluating the Reproducibility of Single-Cell Gene Regulatory Network Inference Algorithms, *Frontiers in Genetics*, *12*, 362, https://doi.org/10.3389/fgene.2021.617282

Katoen, J.-P., (2016), The Probabilistic Model Checking Landscape, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 31-45, https://doi.org/10.1145/2933575.2934574

Kauffman, S., Peterson, C., Samuelsson, B., & Troein, C., (2003), Random Boolean network models and the yeast transcriptional network, *Proceedings of the*

National Academy of Sciences, *100 25*, 14796-14799, https://doi.org/10.1073/pnas.2036429100

Kauffman, S., (1969), Metabolic stability and epigenesis in randomly constructed genetic nets, *Journal of Theoretical Biology*, *22 3*, 437-467, https://doi.org/10.1016/0022-5193(69)90015-0

Khalis, Z., Comet, J.-P., Richard, A., & Bernot, G., (2009), The SMBioNet method for discovering models of gene regulatory networks, *Genes, genomes and genomics*, *3 1*, 15-22.

Kim, H.-E., Kim, H. H., Han, B.-K., Kim, K. H., Han, K., Nam, H., Lee, E. H., & Kim, E.-K., (2020), Changes in cancer detection and false-positive recall in mammography using artificial intelligence : a retrospective, multireader study, *The Lancet Digital Health*, *2 3*, e138-e148, https://doi.org/10.1016/S2589-7500(20)30003-0

Kitano, H., (2002), Systems Biology : A Brief Overview, *Science, New Series*, *295 5560*, 1662-1664, http://www.jstor.org/stable/3075966

Klarner, H., Siebert, H., & Bockmayr, A., (2012a), Time Series Dependent Analysis of Unparametrized Thomas Networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9 5*, 1338-1351, https://doi.org/10.1109/TCBB.2012.61

Klarner, H., Streck, A., Šafránek, D., Kolčák, J., & Siebert, H., (2012b), Parameter Identification and Model Ranking of Thomas Networks, *In* D. Gilbert & M. Heiner (Éd.), *Computational Methods in Systems Biology* (p. 207-226), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-33636-2_13

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M. E., Lönnerberg, P., Furlan, A., et al., (2018), RNA velocity of single cells, *Nature*, *560 7719*, 494-498.

Lähdesmäki, H., Shmulevich, I., & Yli-Harja, O., (2003), On learning gene regulatory networks under the Boolean network model, *Machine learning*, *52 1*, 147-167.

Lee, T. I., (2002), Transcriptional Regulatory Networks in Saccharomyces cerevisiae, *Science*, *298 5594*, 799-804, https://doi.org/10.1126/science.1075090

Legoux, F., Gilet, J., Procopio, E., Echasserieau, K., Bernardeau, K., & Lantz, O., (2019), Molecular mechanisms of lineage decisions in metabolite-specific T cells, *Nature Immunology*, *20 9*, 1244-1255, https://doi.org/10.1038/s41590-019-0465-3

Li, X., Wang, K., Lyu, Y., Pan, H., Zhang, J., Stambolian, D., Susztak, K., Reilly, M. P., Hu, G., & Li, M., (2020), Deep learning enables accurate clustering with batch effect removal in single-cell RNA-seq analysis, *Nature Communications*, *11 1*, 2338, https://doi.org/10.1038/s41467-020-15851-3

Liang, S., Fuhrman, S., & Somogyi, R., (1998), Pacific Symposium on Biocomputing 3 :18-29 (1998), *Pacific Symposium on Biocomputing*, 12.

Lim, C. Y., Wang, H., Woodhouse, S., Piterman, N., Wernisch, L., Fisher, J., & Göttgens, B., (2016), BTR : training asynchronous Boolean models using single-cell expression data, *BMC Bioinformatics*, *171*, https://doi.org/10.1186/s12859-016-1235-y

Liu, Y., Hayes, D. N., Nobel, A., & Marron, J. S., (2008), Statistical Significance of Clustering for High-Dimension, Low–Sample Size Data, *Journal of the American Statistical Association*, *103 483*, 1281-1293, https://doi.org/10.1198/016214508000000454

Longo, D., & Hasty, J., (2006), Dynamics of single-cell gene expression, *Molecular Systems Biology*, *21*, 64, https://doi.org/https://doi.org/10.1038/msb4100110

Luecken, M. D., & Theis, F. J., (2019), Current best practices in single-cell RNA-seq analysis : a tutorial, *Molecular Systems Biology*, *15 6*, https://doi.org/10.15252/msb.20188746

Mandon, H., Su, C., Haar, S., Pang, J., & Paulevé, L., (2019), Sequential Reprogramming of Boolean Networks Made Practical, *In* L. Bortolussi & G. Sanguinetti (Éd.), *Computational Methods in Systems Biology* (p. 3-19), Springer International Publishing, https://doi.org/10.1007/978-3-030-31304-3_1

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., & Stolovitzky, G., (2010), Revealing strengths and weaknesses of methods for gene network inference, *Proceedings of the National Academy of Sciences*, *107 14*, 6286-6291, https://doi.org/10.1073/pnas.0913357107

Martin, S., Zhang, Z., Martino, A., & Faulon, J.-L., (2007), Boolean dynamics of genetic regulatory networks inferred from microarray time series data, *Bioinformatics*, *23 7*, 866-874, https://doi.org/10.1093/bioinformatics/btm021

Martínez, D., Ribeiro, T., Inoue, K., Alenya, G., & Torras, C., (2015), Learning probabilistic action models from interpretation transitions, *Proceedings of the Technical Communications of the 31st International Conference on Logic Programming (ICLP 2015)*, 1-14.

Mateus, D., Gallois, J.-P., Comet, J.-P., & Le Gall, P., (2007), Symbolic modeling of Genetic Regulatory Networks, *Journal of Bioinformatics and Computational Biology*, *5 2B*, 627-640.

Matsumoto, H., Kiryu, H., Furusawa, C., Ko, M. S. H., Ko, S. B. H., Gouda, N., Hayashi, T., & Nikaido, I., (2017), SCODE : an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation, *Bioinformatics*, *33 15*, 2314-2321, https://doi.org/10.1093/bioinformatics/btx194

McInnes, L., Healy, J., & Melville, J., (2018), UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction, *ArXiv e-prints 1802.03426*.

Method of the Year 2013, (2014), *Nature Methods*, *11*, 1, https://doi.org/10.1038/nmeth.2801

Method of the Year 2019 : Single-cell multimodal omics, (2020), *Nature Methods*, *17*, 1, https://doi.org/10.1038/s41592-019-0703-5

Moignard, V., Woodhouse, S., Haghverdi, L., Lilly, A. J., Tanaka, Y., Wilkinson, A. C., Buettner, F., Macaulay, I. C., Jawaid, W., Diamanti, E., Nishikawa, S.-I., Piterman, N., Kouskoff, V., Theis, F. J., Fisher, J., & Göttgens, B., (2015), Decoding the regulatory network of early blood development from single-

cell gene expression measurements, *Nature Biotechnology*, *33 3*, 269-276, https://doi.org/10.1038/nbt.3154

Moraes, F., & Góes, A., (2016), A decade of human genome project conclusion : Scientific diffusion about our genome knowledge : A Decade of Human Genome Project Conclusion, *Biochemistry and Molecular Biology Education*, *44 3*, 215-223, https://doi.org/10.1002/bmb.20952

Mori, F., & Mochizuki, A., (2017), Expected Number of Fixed Points in Boolean Networks with Arbitrary Topology, *Physical Review Letters*, *119 2*, https://doi.org/10.1103/PhysRevLett.119.028301

Muggleton, S., & de Raedt, L., (1994), Inductive Logic Programming : Theory and methods, *The Journal of Logic Programming*, *19-20*, 629-679, https://doi.org/10.1016/0743-1066(94)90035-3

Nabli, F., Martinez, T., Fages, F., & Soliman, S., (2016), On enumerating minimal siphons in Petri nets using CLP and SAT solvers : theoretical and practical complexity, *Constraints*, *21 2*, 251-276, https://doi.org/10.1007/s10601-015-9190-1

Naldi, A., Hernandez, C., Levy, N., Stoll, G., Monteiro, P. T., Chaouiya, C., Helikar, T., Zinovyev, A., Calzone, L., Cohen-Boulakia, S., Thieffry, D., & Paulevé, L., (2018), The CoLoMoTo Interactive Notebook : Accessible and Reproducible Computational Analyses for Qualitative Biological Networks, *Frontiers in Physiology*, *9*, https://doi.org/10.3389/fphys.2018.00680

Noor, A., Serpedin, E., Nounou, M., Nounou, H., Mohamed, N., & Chouchane, L., (2013), An Overview of the Statistical Methods Used for Inferring Gene Regulatory Networks and Protein-Protein Interaction Networks, *Advances in Bioinformatics*, *2013*, 1-12, https://doi.org/10.1155/2013/953814

Ogata, S., Tsuchiya, T., & Kikuno, T., (2004), SAT-Based Verification of Safe Petri Nets, *In* F. Wang (Éd.), *Automated Technology for Verification and Analysis* (p. 79-92), Springer Berlin Heidelberg.

Paulevé, L., (2017), Pint : A Static Analyzer for Transient Dynamics of Qualitative Networks with IPython Interface, *In* J. Feret & H. Koeppl (Éd.), *Computational*

Methods in Systems Biology (p. 309-316), Springer International Publishing, https://doi.org/10.1007/978-3-319-67471-1_20

Paulevé, L., (2018), Reduction of Qualitative Models of Biological Networks for Transient Dynamics Analysis, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *15 4*, 1167-1179, https://doi.org/10.1109/TCBB.2017.2749225

Paulevé, L., (2020), *Réseaux booléens : méthodes formelles et outils pour la modélisation en biologie* (Habilitation à diriger des recherches), Université Paris-Saclay, https://tel.archives-ouvertes.fr/tel-03150976

Paulevé, L., Magnin, M., & Roux, O., (2011), Refining Dynamics of Gene Regulatory Networks in a Stochastic π-Calculus Framework, *In* C. Priami, R.-J. Back, I. Petre & E. de Vink (Éd.), *Transactions on Computational Systems Biology XIII* (p. 171-191), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-19748-2_8

Paulevé, L., Magnin, M., & Roux, O., (2012), Static analysis of Biological Regulatory Networks dynamics using abstract interpretation, *Mathematical Structures in Computer Science*, *22 4*, 651-685, https://doi.org/10.1017/S0960129511000739

Paulevé, L., Kolčák, J., Chatain, T., & Haar, S., (2020), Reconciling qualitative, abstract, and scalable modeling of biological networks, *Nature Communications*, *11 1*, 4256, https://doi.org/10.1038/s41467-020-18112-5

Plaisted, D. A., & Greenbaum, S., (1986), A Structure-preserving Clause Form Translation, *Journal of Symbolic Computation*, *2 3*, 293-304, https://doi.org/10.1016/S0747-7171(86)80028-1

Ponty, Y., (2020), *Ensemble Algorithms and Analytic Combinatorics in RNA Bioinformatics and Beyond* (Habilitation à diriger des recherches), Université Paris-Saclay, https://tel.archives-ouvertes.fr/tel-03219977

Potter, S. S., (2018), Single-cell RNA sequencing for the study of development, physiology and disease, *Nature Reviews Nephrology*, *14 8*, 479-492, https://doi.org/10.1038/s41581-018-0021-7

Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A., & Murali, T. M., (2019), Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data, *bioRxiv*, https://doi.org/10.1101/642926

Qiu, M., Khisamutdinov, E., Zhao, Z., Pan, C., Choi, J.-W., Leontis, N. B., & Guo, P., (2013), RNA nanotechnology for computer design and *in vivo* computation, *Philosophical Transactions of the Royal Society A : Mathematical, Physical and Engineering Sciences*, *371 2000*, 20120310, https://doi.org/10.1098/rsta.2012.0310

Ribeiro, T., Magnin, M., Inoue, K., & Sakama, C., (2015), Learning Multi-valued Biological Models with Delayed Influence from Time-Series Observations, *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 25-31, https://doi.org/10.1109/ICMLA.2015.19

Ribeiro, T., Tourret, S., Folschette, M., Magnin, M., Borzacchiello, D., Chinesta, F., Roux, O., & Inoue, K., (2018a), Inductive Learning from State Transitions over Continuous Domains, *In* N. Lachiche & C. Vrain (Éd.), *Inductive Logic Programming* (p. 124-139), Springer International Publishing.

Ribeiro, T., Folschette, M., Magnin, M., Roux, O., & Inoue, K., (2018b), Learning Dynamics with Synchronous, Asynchronous and General Semantics, *In* F. Riguzzi, E. Bellodi & R. Zese (Éd.), *Inductive Logic Programming* (p. 118-140), Springer International Publishing, https://doi.org/10.1007/978-3-319-99960-9_8

Ribeiro, T., Folschette, M., Magnin, M., & Inoue, K., (2021), Polynomial Algorithm For Learning From Interpretation Transition, *1st International Joint Conference on Learning & Reasoning*, https://hal.archives-ouvertes.fr/hal-03347026

Ribeiro, T., Folschette, M., Trilling, L., Glade, N., Inoue, K., Magnin, M., & Roux, O., (to appear), *Les enjeux de l'inférence de modèles dynamiques des systèmes biologiques à partir de séries temporelles*, https://hal.archives-ouvertes.fr/hal-02634235

Ruhe, G., (1988), Parametric maximal flows in generalized networks – complexity and algorithms, *Optimization*, *19 2*, 235-251, https://doi.org/10.1080/02331938808843341

Saelens, W., Cannoodt, R., Todorov, H., & Saeys, Y., (2019), A comparison of single-cell trajectory inference methods, *Nature Biotechnology*, *37 5*, 547-554, https://doi.org/10.1038/s41587-019-0071-9

Schwab, J. D., Kühlwein, S. D., Ikonomi, N., Kühl, M., & Kestler, H. A., (2020), Concepts in Boolean network modeling : What do they all mean ?, *Computational and Structural Biotechnology Journal*, *18*, 571-582, https://doi.org/10.1016/j.csbj.2020.03.001

Schwab, J. D., Ikonomi, N., Werle, S. D., Weidner, F. M., Geiger, H., & Kestler, H. A., (2021), Reconstructing Boolean network ensembles from single-cell data for unraveling dynamics in the aging of human hematopoietic stem cells, *Computational and Structural Biotechnology Journal*, *19*, 5321-5332, https://doi.org/10.1016/j.csbj.2021.09.012

Schwartzkopff, F., Petersen, F., Grimm, T. A., & Brandt, E., (2012), CXC chemokine ligand 4 (CXCL4) down-regulates CC chemokine receptor expression on human monocytes, *Innate Immun*, *18 1*, 124-139.

Sheikh, I. A., Ahmad, J., Magnin, M., & Roux, O., (2019), Incorporating Time Delays in Process Hitting Framework for Dynamical Modeling of Large Biological Regulatory Networks, *Frontiers in Physiology*, *10*, 90, https://doi.org/10.3389/fphys.2019.00090

Shen-Orr, S. S., Tibshirani, R., Khatri, P., Bodian, D. L., Staedtler, F., Perry, N. M., Hastie, T., Sarwal, M. M., Davis, M. M., & Butte, A. J., (2010), Cell type–specific gene expression differences in complex tissues, *Nature methods*, *7 4*, 287-289.

Shmulevich, I., Dougherty, E. R., Kim, S., & Zhang, W., (2002), Probabilistic Boolean networks : a rule-based uncertainty model for gene regulatory networks, *Bioinformatics*, *18 2*, 261-274, https://doi.org/10.1093/bioinformatics/18.2.261

Specht, A. T., & Li, J., (2016), LEAP : constructing gene co-expression networks for single-cell RNA-sequencing data using pseudotime ordering, *Bioinformatics*, btw729, https://doi.org/10.1093/bioinformatics/btw729

Stoll, G., Viara, E., Barillot, E., & Calzone, L., (2012), Continuous time boolean modeling for biological signaling : application of Gillespie algorithm, *BMC Systems Biology*, *6 1*, 116, https://doi.org/10.1186/1752-0509-6-116

Student, (1908), The Probable Error of a Mean, *Biometrika*, *6 1*, 1-25, https://doi.org/10.1093/biomet/6.1.1

the Mutation Consequences and Pathway Analysis working group of the International Cancer Genome Consortium, (2015), Pathway and network analysis of cancer genomes, *Nature Methods*, *12 7*, 615-621, https://doi.org/10.1038/nmeth.3440

The Respiratory Viral DREAM Challenge Consortium, Fourati, S., Talla, A., Mahmoudian, M., Burkhart, J. G., Klén, R., Henao, R., Yu, T., Aydin, Z., Yeung, K. Y., Ahsen, M. E., Almugbel, R., Jahandideh, S., Liang, X., Nordling, T. E. M., Shiga, M., Stanescu, A., Vogel, R., Pandey, G., ... Sieberts, S. K., (2018), A crowdsourced analysis to identify ab initio molecular signatures predictive of susceptibility to viral infection, *Nature Communications*, *9 1*, 4418, https://doi.org/10.1038/s41467-018-06735-8

Thieffry, D., & Thomas, R., (1995), Dynamical behaviour of biological regulatory networks—II. Immunity control in bacteriophage lambda, *Bulletin of Mathematical Biology*, *57 2*, 277-297, https://doi.org/10.1016/0092-8240(94)00037-D

Thierry-Mieg, Y., (2015), Symbolic Model-Checking Using ITS-Tools, *In* C. Baier & C. Tinelli (Éd.), *Tools and Algorithms for the Construction and Analysis of Systems* (p. 231-237), Springer Berlin Heidelberg.

Thomas, R., (1981), On the Relation Between the Logical Structure of Systems and Their Ability to Generate Multiple Steady States or Sustained Oscillations, *In* J. Della Dora, J. Demongeot & B. Lacolle (Éd.), *Numerical Methods in the Study of Critical Phenomena* (p. 180-193), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-81703-8_24

Thomas, R., (1973), Boolean formalization of genetic control circuits, *Journal of Theoretical Biology*, *42 3*, 563-585, https://doi.org/https://doi.org/10.1016/0022-5193(73)90247-6

Thomas, R., Thieffry, D., Kaufman, M., & de Chimie-Physique, S., (1995), Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state, *Bulletin of Mathematical Biology*, *57*, 247-276, https://doi.org/10.1007/BF02460618

Thorne, T., (2018), Approximate inference of gene regulatory network models from RNA-Seq time series data, *BMC Bioinformatics*, *19 1*, 127, https://doi.org/10.1186/s12859-018-2125-2

Tseitin, G. S., (1983), On the Complexity of Derivation in Propositional Calculus, *In* J. H. Siekmann & G. Wrightson (Éd.), *Automation of Reasoning : 2 : Classical Papers on Computational Logic 1967–1970* (p. 466-483), Springer Berlin Heidelberg, https://doi.org/10.1007/978-3-642-81955-1_28

Turing, A. M., (1950), I.—COMPUTING MACHINERY AND INTELLIGENCE, *Mind*, *59 236*, 433-460, https://doi.org/10.1093/mind/LIX.236.433

Turing, A., (1952), The chemical basis of morphogenesis, *Philosophical Transactions of the Royal Society B*, *237*, 37-72.

Van der Maaten, L., & Hinton, G., (2008), Visualizing data using t-SNE., *Journal of machine learning research*, *9 11*.

Van Giang, T., & Hiraishi, K., (2020), An efficient method for approximating attractors in large-scale asynchronous Boolean models, *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 1820-1826.

Wang, R.-S., Saadatpour, A., & Albert, R., (2012), Boolean modeling in systems biology : an overview of methodology and applications, *Physical Biology*, *9 5*, 055001, https://doi.org/10.1088/1478-3975/9/5/055001

Wichmann, B., Canning, A., Marsh, D., Clutterbuck, D., Winsborrow, L., & Ward, N., (1995), Industrial perspective on static analysis, *Software Engineering Journal*, *10 2*, 69, https://doi.org/10.1049/sej.1995.0010

Wolf, F. A., Angerer, P., & Theis, F. J., (2018), SCANPY : large-scale single-cell gene expression data analysis, *Genome biology*, *19 1*, 1-5.

Woodhouse, S., Piterman, N., Wintersteiger, C. M., Göttgens, B., & Fisher, J., (2018), SCNS : a graphical tool for reconstructing executable regulatory networks from single-cell genomic data, *BMC Systems Biology*, *12 1*, https://doi.org/10.1186/s12918-018-0581-y

Yi Ming Zou, (2013), An algorithm for detecting fixed points of boolean network, *2013 ICME International Conference on Complex Medical Engineering*, 670-673, https://doi.org/10.1109/ICCME.2013.6548334

Zhang, S.-Q., Hayashida, M., Akutsu, T., Ching, W.-K., & Ng, M. K., (2007), Algorithms for Finding Small Attractors in Boolean Networks, *EURASIP Journal on Bioinformatics and Systems Biology*, *2007*, 1-13, https://doi.org/10.1155/2007/20180

Titre : Vérification formelle et apprentissage logique pour la modélisation qualitative à partir de données *single-cell*

Mot clés : biologie des systèmes, réseaux de régulation génétique, séquençage *single-cell*, modèles qualitatifs, méthodes formelles, apprentissage automatique

Résumé : La compréhension des mécanismes cellulaires à l'œuvre au sein des organismes vivants repose généralement sur l'étude de leur expression génétique. Cependant, les gènes sont impliqués dans des processus de régulation complexes et leur mesure est difficile à réaliser. Dans ce contexte, la modélisation qualitative des réseaux de régulation génétique vise à établir la fonction de chaque gène à partir de la modélisation discrète d'un réseau d'interaction dynamique. Dans cette thèse, nous avons pour objectif de mettre en place cette approche de modélisation à partir des données de séquençage *single-cell*. Ces données se révèlent en effet intéressantes pour la modélisation qualitative, car elles apportent une grande précision et peuvent être interprétées de manière dynamique. Nous développons ainsi une méthode d'inférence de modèles qualitatifs basée sur l'apprentissage automatique de programmes logiques. Cette méthode est mise en œuvre sur des données *single-cell* et nous proposons plusieurs approches pour interpréter les modèles résultants en les confrontant avec des connaissances préétablies.

Title: Formal verification and automatic learning of logic programs for qualitative modeling with single-cell data

Keywords: systems biology, genetic regulatory networks, single-cell sequencing, qualitative models, formal methods, machine learning

Abstract: The understanding of cellular mechanisms occurring inside human beings usually depends on the study of its gene expression. However, genes are implied in complex regulatory processes and their measurement is difficult to perform. In this context, the qualitative modeling of gene regulatory networks intends to establish the function of each gene from the discrete modeling of a dynamical interaction network. In this thesis, our goal is to implement this modeling approach from single-cell sequencing data. These data prove to be interesting for qualitative modeling since they bring high precision, and they can be interpreted in a dynamical way. Thus, we develop a method for the inference of qualitative models based on the automatic learning of logic programs. This method is applied on a single-cell dataset, and we propose several approaches to interpret the resulting models by comparing them with existing knowledge.