Génération de prouveurs automatiques pour la logique modale avec PGeon

Mémoire de fin d'étude

Master *Sciences et Technologies*,
Mention *Informatique*,
Parcours IASD, CMI, MTS

Auteur
PIERRET Matthieu

Superviseurs
DELAHAYE David
ROBILLARD Simon
SIDHOUM Romain

Lieu de stage
LIRMM, Montpellier, France

version 1 soutenu publiquement le 29 juin 2026

Résumé

Ce stage académique se situe dans le domaine de la démonstration automatique et porte sur une extension de PGeon, un outil de génération de prouveurs automatiques (ATP). Le développement d'ATPs est spécifique à chaque logique et est une tâche coûteuse, ce qui limite la réutilisation des outils. PGeon répond à cette problématique en générant des moteurs de preuve basés sur la méthode des tableaux sémantiques à partir d'une simple description formelle de la logique cible.

L'objectif principal de ce travail est d'implémenter une modélisation de la logique modale S4 au sein de PGeon. Jusqu'ici, le travail a consisté en une veille bibliographique, puis à adapter un calcul des séquents propositionnel pour le transformer en règles exploitables par le moteur de recherche de PGeon. Cette approche permet de générer des prouveurs pour traiter des systèmes complexes.

Les travaux à venir consisteront à bien étudier la logique modale S4 que nous allons modéliser dans Pgeon. En particulier, nous nous poserons la question de savoir si nous optons pour une logique classique ou intuitionniste (les deux systèmes existent). Nous regarderons également si les variantes de la logique modale S4, ainsi que les systèmes de preuve associés, possèdent de bonnes propriétés comme la décidabilité, l'élimination des coupures, etc. Enfin, une fois la version propositionnelle modélisée, nous nous proposons également de regarder son extension au premier ordre.

Abstract

This academic internship is set within the field of automated reasoning and focuses on an extension of PGeon, a framework for generating Automated Theorem Provers (ATP). The development of ATPs is typically specific to each logic and remains a laborious task, which limits the interoperability and reuse of such tools. PGeon addresses this challenge by generating proof engines based on the semantic tableau method directly from a formal descriptive specification of the target logic.

The primary objective of this work is to implement a model of S4 modal logic within PGeon. To date, the project has involved an extensive bibliographic review, followed by the adaptation of a propositional sequent calculus into rules compatible with PGeon's search engine. This approach enables the generation of provers capable of handling complex systems.

Future work will focus on a detailed study of the S4 modal logic to be modeled in PGeon. In particular, we will investigate whether to opt for a classical or an intuitionistic foundation, as both systems are available. We will also examine whether the variants of S4, and their associated proof systems, possess key properties such as decidability and cut-elimination. Finally, once the propositional version is modeled, we propose to explore its extension to first-order logic.

Remerciements

Je voudrais commencer par remercier David Delahaye, qui est mon maître de stage au LIRMM, mais également un professeur qui a pu m'aider depuis 5 ans, tant dans mes projets professionnels que dans certains projets personnels. Avec lui je remercie Julie Cailler, et la félicite pour son poste de MCF à l'université de Nancy, merci pour le stage de L2, qui a été charnière dans mon parcours.

Je remercie bien sûr toute l'équipe enseignante du département informatique, merci de m'avoir donné la motivation et les ressources pour aller plus loin. Merci aussi à MM. Christian Rétoré et Simon Robillard, ainsi qu'aux autres membres de mon équipe de recherche au LIRMM, M. Moot, Romain, Loïc, Léo, Vincent, Clémence et les nouveaux stagiaires. Merci à Anne-Élisabeth Baert et Sébastien Da-Silva, pour la gestion du CMI, bien que ce ne fût pas toujours facile ! Merci à Clément et Lucas, ainsi qu'aux quelques survivants de cette promotion CMI, bien que nous soyons bien plus nombreux que les anciennes promos ! Merci et MM. Leclère et Sioutis pour les retours sur le rendu préliminaire.

Merci également à mes amis sans qui l'aventure aurait été bien moins agréable. Que ce soit mes camarades de classe, avec qui je m'entendais en grande majorité bien, ou ceux qui sont partis dans d'autres masters ou arrêté leurs études. Merci beaucoup Antoine et Théo. Merci Noah et Isaac. Je remercierai Oscar sur Monifactory. Merci à Furmafu.

C'était un plaisir de m'investir dans la vie étudiante, d'avoir participé à tant de projets et d'actions bénévoles. J'ai beaucoup appris, et ce sont des expériences que je garderai toute ma vie, malgré parfois quelques points négatifs inévitables.

Table des matières

Table des figures

1

Introduction

Les prouveurs automatiques sont des programmes qui, à partir d'axiomes et de règles d'inférence, cherchent une preuve formelle d'un énoncé logique. Ils explorent un espace de preuves potentiellement grand en appliquant des règles à chaque étape, en utilisant des heuristiques pour maximiser les performances. Ces programmes sont au coeur de domaines comme la vérification de programmes critiques, les mathématiques assistées par ordinateur, ou encore l'intelligence artificielle (symbolique).

Le développement de prouveurs automatiques est un travail qui prend souvent beaucoup de temps, et où chacun de ces programmes définit la syntaxe d'une logique et une heuristique spécifique, ce qui nuit à la réutilisation des outils. *PGeon* est un outil qui a justement pour but de générer des prouveurs automatiques à partir d'un fichier descriptif de la syntaxe et des règles d'inférences de la logique à implémenter. L'objectif de ce stage est d'implémenter une modélisation d'une logique modale dans *PGeon*, et de fournir un banc de tests complet avec une analyse des résultats.

Les logiques modales sont une extension d'un système logique avec un opérateur exprimant la nécessité $\Box$ et la possibilité $\Diamond$. Cette sémantique autorise le raisonnement à se porter sur des mondes possibles parallèles, où les propriétés sont relatives à ces mondes. Cela peut représenter les états d'un système, les croyances d'une population, ou les connaissances et interactions d'agents.

Dans PGeon, plusieurs logiques et méthodes de recherche de preuve ont été modélisées (logiques propositionnelles classique et intuitionniste, premier ordre...). En modélisant les logiques modales, ce stage permettra ainsi de mesurer la polyvalence de l'outil.

2

Pgeon

Pgeon est un outil générique de génération de prouveurs automatiques (ATP), développé par Romain Sidhoum dans le cadre de sa thèse en informatique, réalisée au LIRMM. Au lieu de programmer un ATP à la main pour chaque nouvelle logique, on décrit dans Pgeon la syntaxe de la logique visée, donc les formes des formules, les symboles, ainsi que les règles d'inférence (prémisses, conclusion, conditions d'application, etc.), et enfin la stratégie de contrôle de l'application de ces règles (l'ordre d'exploration les priorités, le backtracking). Ensuite l'outil se charge de produire automatiquement un moteur de recherche de preuve correspondant. L'idée est donc de factoriser l'implémentation de la partie moteur, comme la recherche ou la gestion des branches, pour se concentrer sur la définition formelle de la logique elle-même.

L'intérêt d'avoir un générateur de prouveurs automatique repose dans le fonctionnement de ces prouveurs. Il existe déjà une grande variété d'ATPs, chacun implémentant la syntaxe d'une logique et des stratégies particulières, les rendant plus ou moins efficaces sur ces problèmes. Cependant, ces stratégies ou logiques implémentées limitent le potentiel de réutilisation de ces prouveurs. Ainsi, en générer en décrivant uniquement les stratégies à suivre devient très intéressant, la réutilisation étant facilitée.

La logique modale n'est pas un domaine où les théories sont autant mises en pratique que pour la logique du premier ordre par exemple. Seulement quelques outils similaires existent déjà, comme *MetTeL*^{2} [STK$^+$12], *gen2sat* ou *Tableaux Workbench* [AG09]. Mais contrairement à PGeon, ces outils sont limités à des logiques propositionnelles, alors que PGeon peut traiter des logiques quantifiées. De la même manière que *MetTeL*^{2} et *Tableaux workbench*, les ATPs générés par *PGeon* sont basés sur la méthode des tableaux sémantiques, une méthode de preuve répandue pour les logiques du premier ordre ou modales.

Tableaux sémantiques

Les tableaux sémantiques sont une méthode de preuve pour certaines logiques développée originellement en 1955 par le logicien Evert Willem Beth. Elle a ensuite été peaufinée par le philosophe Jaako Hintikka avant d'être formalisée par Raymond Smullyan. C'est d'ailleurs parmi les méthodes les plus utilisées en logique modale, et c'est cette tech-

nique qu'implémente nativement *PGeon*. Pour démontrer une formule, le principe est de montrer que sa négation est insatisfiable.

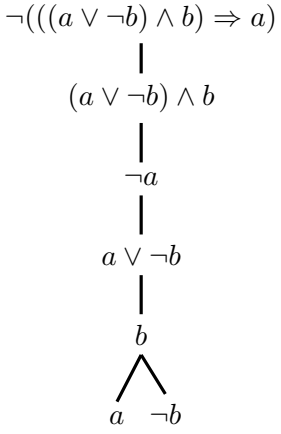
La méthode se repose sur un arbre de preuve se lisant du haut vers le bas. Les règles peuvent soit ajouter une formule, soit séparer l'arbre en deux (brancher) avant d'ajouter une formule dans chacune des branches. Le but est de prouver une formule en montrant que sa négation est insatisfiable. Dans les règles, la situation initiale est la formule au sommet de l'arbre, et les conséquences sont le reste. Il existe en plus la règle de clôture : la présence d'une formule et de sa négation dans la même branche clôt la branche. Une branche close est insatisfiable, ainsi un arbre ne consistant que en branches closes est notre objectif avec cette méthode. Ces outils et méthodes sont très utiles, particulièrement dans la modélisation de systèmes critiques, où les propriétés se doivent d'être vérifiées. Ces règles se présentent sous la forme :

$$\frac{\text{prémisses}}{\text{conclusion}}\ \text{Nom de la règle}$$

$$\frac{X\ ;\ Y}{X}\,(\theta) \qquad \frac{X\ ;\ P\ ;\ \neg P}{\bot}\,(\bot) \qquad \frac{X\ ;\ \neg\neg P}{X\ ;\ P}\,(\neg)$$

$$\frac{X\ ;\ P \wedge Q}{X\ ;\ P\ ;\ Q}\,(\wedge) \qquad \frac{X\ ;\ P \vee Q}{X\ ;\ P \quad | \quad X\ ;\ Q}\,(\vee)$$

FIGURE 2.1 : Règles tableaux pour LK propositionnel



Si l'on veut prouver la formule $((a \vee \neg b) \wedge b) \Rightarrow a$, alors il faut commencer l'arbre par la négation de la formule. Si l'on peut montrer que sa négation est insatisfiable (donc que toutes les branches se ferment), alors on peut en déduire que la formule initiale est valide.
Ici je commence par appliquer $\neg(A \Rightarrow B)$, ce qui nous produit $(a \vee \neg b) \wedge b$ ainsi que $\neg a$, les deux dans le même branche. $(a \vee \neg b) \wedge b$ produit $a \vee \neg b$ et b dans la même branche. Et enfin, $a \vee \neg b$ produit une branche avec a et une avec $\neg b$.
Maintenant, chaque branche se ferme correctement, la branche contenant les 4 premières formules et a se ferme avec a et $\neg a$, et celle contenant le 4 premières et b se ferme avec b et $\neg b$. Ainsi, la formule initiale est valide.

Encodage

L'implémentation dans *PGeon* se fait par la création de fichiers textes, formatés avec des mots clefs afin de définir la logique du prouveur à générer. Il faut ainsi modéliser les règles, décrivant les transformations syntaxiques à effectuer en cas d'application d'une règle.

```
main type formula

function not  : formula -> formula
function and  : formula formula -> formula
function or   : formula formula -> formula
function impl : formula formula -> formula

rule rAxiom   : P ; not(P) ==X

rule rAnd     : and(P, Q)        ==> P ; Q
rule rOr      : or(P, Q)         ==> P | Q
rule rImpl    : impl(P, Q)       ==> not(P) | Q
rule rNotNot  : not(not(P))      ==> P
rule rNotAnd  : not(and(P, Q))   ==> not(P) | not(Q)
rule rNotOr   : not(or(P, Q))    ==> not(P) ; not(Q)
rule rNotImpl : not(impl(P, Q)) ==> P ; not(Q)

strategy closure : rAxiom
strategy alpha : rAnd || rNotNot || rNotOr || rNotImpl
strategy beta : rOr || rImpl || rNotAnd
main strategy prove : (closure || alpha || beta)*
```

Listing 2.1: Définition de LK propositionnel dans PGeon

La logique descriptive se sépare en trois parties, la partie déclarative, suivie de la partie de règles, et enfin la partie de stratégie. La partie déclarative consiste à exprimer les types et la forme des objets manipulables. Les deux suivantes sont plus intéressantes d'un point de vue implémentation. Tout d'abord, il faut comprendre la sémantique des symboles pour la déclaration de règles et stratégies 2.2. La partie de règles est l'ensemble des règles syntaxiques de réécriture pour avancer dans la recherche de preuve. Dans la définition de LK propositionnel dans PGeon, les règles décrites sont simplement les règles tableaux 2.1. La stratégie nécessite plus d'explication, l'exemple, bien que simple, sera plus parlant que la définition formelle des opérateurs. Tout d'abord on a créé trois ensembles de règles *closure*($\odot$), *alpha*(α) et *beta*(β), l'idée derrière est leur place dans la recherche de preuve. Les règles $\odot$ terminent la recherche de preuve dans une branche, les règles α détruisent un connecteur pour ajouter des formules à la branche, et les règles β détruisent un connecteur mais créent deux branches. Ainsi, l'intuition pour minimiser l'espace de recherche de preuve est d'effectuer les transformations les plus simples avant de brancher, afin d'éviter de multiplier l'espace. C'est exactement ce qu'indique la stratégie de recherche de preuve (indiquée par le mot clef *main*) : mettre les trois ensembles de règles à la suite par des opérateurs biaisés à gauche, le tout avec une étoile de Kleene pour trouver la preuve en 0, 1 ou n étapes. Il s'agit donc d'essayer d'appliquer *rAxiom*, si ce n'est pas possible, on essaye *rAnd*, puis *rNotNot*, etc.

Symbole	Type	Sémantique
Γ ; Δ	règle	Les formules de Γ et Δ sont sur la même branche.
$\Gamma \mid \Delta$	règle	Les formules de Γ et Δ sont sur deux branches différentes.
Γ ==X	règles	Les formules de Γ permettent de fermer la branche.
Γ ==> Δ	règle	Γ se réécrit en Δ par règle syntaxique inversible.
Γ ——> Δ	règle	Γ se réécrit en Δ par règle syntaxique non-inversible.
$(\Gamma; ...B)\|...T$ ——> $(\Delta; ...B)\|...T$	règle	La branche contenant Γ se réécrit en la branche contenant Δ, en réécrivant Γ en Δ.
$A\,;B$	stratégie	Composition séquentielle, application de A sur un état s puis de B sur tous les états résultants. $[A;B](s) \equiv [B] >>= [A](s)$.
$A \,\|\|B$	stratégie	Choix biaisé à gauche, application de A si $[A](s) \neq \emptyset$, B sinon.
$A\,\&;B$	stratégie	Composition séquentielle équitable. Similaire à la composition classique, mais si l'une des branches générées par A échoue à appliquer B, l'évaluation globale n'est pas bloquée.
$A \,\&\&B$	stratégie	Intersection. Applique A et B en parallèle sur l'état courant et ne conserve que les états résultants communs aux deux stratégies ($[A\&\&B](s) \equiv [A](s) \cap [B](s)$).
A^*	stratégie	Étoile de Kleene (Répétition). Applique la stratégie A de manière itérative jusqu'à ce qu'elle ne produise plus aucun nouvel état ($\emptyset \,\&\, [A](t) \,\&\, [A] >>= [A](t) \,\&\, ...$).
try(A)	stratégie	Tentative d'application. Équivalent à $[A](s) \,\|\|\, s$, tente d'appliquer A, et si A échoue, retourne l'état initial intact au lieu de lever un échec.

FIGURE 2.2 : Tableau sémantique des symboles

3

Logiques modales

3.1 Logique modale

La logique modale est une forme de logique formelle qui se voit étendre les logiques propositionnelles, ou d'ordre 1 et supérieur, en y ajoutant des modalités. Lesdites modalités représentent la nécessité $\Box$ et la possibilité $\Diamond$ [BRV01, Gar24]. Ces modalités fonctionnent de manière duale :

$$\neg\Box\phi \equiv \Diamond\neg\phi$$

$$\texttt{Phi n'est pas nécessaire} \equiv \texttt{Il est possible que non phi}$$

$$\neg \Diamond \phi \equiv \Box\neg\phi$$

$$\texttt{Il n'est pas possible que phi} \equiv \texttt{Il est nécessaire que non phi}$$

Idéalement, nous préférerions utiliser un socle de logique classique, puisque *PGeon* est un outil à destination de mathématiciens pour aider à la génération de prouveurs (lesquels vérifient les preuves), et que la majorité des mathématiques et mathématiciens modernes travaillent avec la logique classique. La différence majeure entre la logique classique et l'intuitionniste est la présence ou non de l'axiome du tiers exclu : $a \vee \neg a$. Cependant, il existe des ponts entre la logique intuitionniste propositionnelle et S4 modal intuitionniste, et nos recherches sont toujours en cours pour décider de la base définitive du projet, pour l'instant classique.

3.2 Sémantique de Kripke

Un modèle de Kripke est une structure utilisée en logique modale afin d'évaluer la valeur de vérité de formules dans des mondes possibles. Ils permettent en effet d'avoir des propositions, comme "a", vraies dans un monde, et fausses dans un autre. Les mondes peuvent également être mis en relation, afin de s'intéresser aux modalités. Cette modélisation permet de traiter de nombreux problèmes, notamment avec des logiques de description, des logiques pour des agents ou des logiques temporelles.

Un modèle de Kripke est un triplet $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \Vdash \rangle$ où $\mathcal{W}$ est l'ensemble des mondes possibles, $\mathcal{R}$ une relation d'accessibilité binaire entre les mondes, et $\Vdash_0$ une relation "force" entre mondes et formules. Soit $w_1, w_2 \in \mathcal{W}^2$: $w_1 \mathcal{R} w_2$ signifie alors que w_2 est accessible depuis w_1. La formule $w_1 \Vdash \phi$ se dit w_1 force ϕ, ce qui signifie que ϕ est satisfiable dans w_1. On dit que ϕ est valide dans $\mathcal{M}$, défini par $\mathcal{M} \models \phi$, si et seulement si $\forall w.w \in \mathcal{W} \Rightarrow w \Vdash \phi$. On dit que ϕ est satisfiable dans $\mathcal{M}$ si $\exists w \in \mathcal{W}.w \Vdash \phi$. Également, $w \Vdash \neg\phi \Leftrightarrow w \nVdash \phi$. Soient ϕ_1 et ϕ_2 deux formules sans modalités :

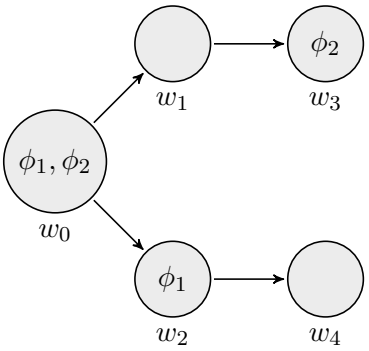


FIGURE 3.1 : Exemple d'un modèle de Kripke avec K

Dans ce modèle de Kripke, nous avons $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \Vdash \rangle$, avec :

- $\mathcal{W} = \{w_0, w_1, w_2, w_3, w_4\}$
- $\mathcal{R} = \{w_0.w_1, w_0.w_2, w_1.w_3, w_2.w_4\}$
- $\Vdash = \{w_0.\phi_1, w_0.\phi_2, w_2.\phi_1, w_3.\phi_2\}$

Les modalités $\Box$ et $\Diamond$ sont alors définies comme ceci :

$$w \Vdash \Box\phi \Leftrightarrow \forall w'.w\mathcal{R}w' \rightarrow w' \Vdash \phi$$

$$w \Vdash \Diamond\phi \Leftrightarrow \exists w'.w\mathcal{R}w' \rightarrow w' \Vdash \phi$$

Ainsi, dans le modèle de Kripke de la figure 3.1, voici quelques affirmations :

- $w_0 \Vdash \phi_1 \rightarrow \phi_2$, puisque $w_0 \Vdash \phi_1$ et $w_0 \Vdash \phi_2$,
- $w_0 \Vdash \Diamond\phi_1$, puisque $w_0 \mathcal{R} w_2$ et $w_2 \Vdash \phi_1$,
- $w_1 \Vdash \Box\phi_2$, puisque tous les mondes accessibles depuis w_1 forcent ϕ_2,
- $w_4 \Vdash \Box\phi_1$, même justification qu'au dessus (par vacuité).

3.3 Axiomatique

Il est possible d'agrémenter les logiques modales de différents axiomes, ce qui permet à différents systèmes d'exister [Gar24]. Voici les principaux axiomes des logiques modales, avec leur impact sur la relation d'accessibilité.

$$\text{Nec } \frac{A}{\Box A} \qquad\qquad \text{MP } \frac{A \qquad A \Rightarrow B}{B}$$

FIGURE 3.2 : Règles de nécessitation et du Modus Ponens

Nom	**Axiome**	**Relation d'accessibilité (R)**
K	$\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$	Axiome de base
D	$\Box P \Rightarrow \Diamond P$	Sérielle
T	$P \Rightarrow \Diamond P$	Réflexive
B	$P \Rightarrow \Box\Diamond P$	Symétrique
4	$\Box P \Rightarrow \Box\Box P$	Transitive
5	$\Diamond P \Rightarrow \Box\Diamond P$	Relation d'équivalence

TABLE 3.1 : Axiomes de la logique modale et propriétés de la relation d'accessibilité

Les intuitions pour ces axiomes sont les suivantes : K exprime que s'il est nécessaire que A implique B, et que s'il est nécessaire que A, alors il est aussi nécessaire que B. D dit que la nécessité de P implique sa possibilité, et oblige l'existence d'au moins un monde forçant P, ce qui rend $\mathcal{R}$ sériel. T que l'existence de P implique sa possibilité, cela implique la réflexivité de $\mathcal{R}$. B exprime que l'existence de P implique la nécessité de sa possibilité, rendant $\mathcal{R}$ symétrique. Enfin, *4* assure la transitivité dans tous les mondes, et *5* implique que l'accessibilité soit une relation d'équivalence (réflexive, transitive et symétrique).

Il existe donc de nombreuses façons d'ajouter des éléments à ces modèles de Kripke en agrémentant la relation d'accessibilité de certaines propriétés, voici quelques logiques modales très utilisées (axiomes en italique et logiques en gras) :

K : K + *Nec* + *MP*

T : **K** + T

D : **K** + D

S4 : **T** + *4*

S5 : **T** + *5*

Le lien entre le choix des axiomes et la sémantique de Kripke n'est pas forcément évident, il faut s'appuyer sur la définition sémantique des modalités. Notre exemple initial, la figure 3.1 ne se reposait que sur le système **K**, mais si l'on décide de changer

pour **S4**, en gardant les cinq mondes possibles et avec $w_0 \Vdash \phi_1$ et $w_0 \Vdash \phi_2$, un modèle de Kripke serait :

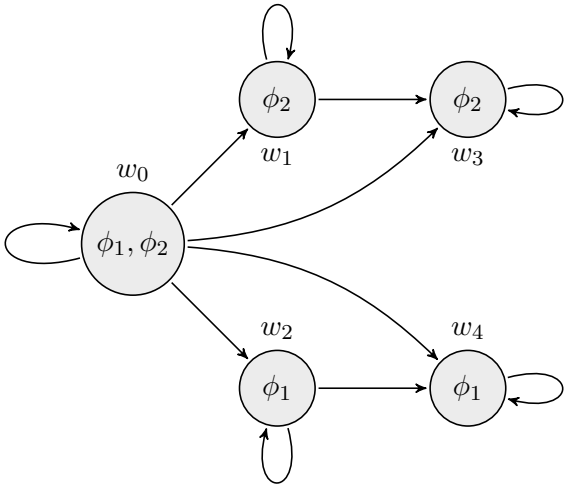


FIGURE 3.3 : Modèle de Kripke pour *S4*

Le système S4 nous garantit bien les propriétés de réflexivité et transitivité pour la relation d'accessibilité. Nous avons décidé de partir sur la logique modale S4, pour ces raisons, ainsi que :

- **Propriétés de réduction modale :** La combinaison de la réflexivité et de la transitivité (T+4) permet de borner l'espace des modalités distinctes à un ensemble fini (réduction des séquences de boîtes et de diamants). Cette propriété est un atout pour des stratégies de recherche de preuves terminantes.
- **Expressivité et domaines d'application :** En modélisant une relation d'accessibilité s'apparentant à un préordre, S4 constitue la base pour les logiques épistémiques et temporelles. C'est un candidat idéal pour évaluer la polyvalence et l'expressivité du générateur des prouveurs générés.
- **Interopérabilité avec la logique intuitionniste :** Il existe des ponts syntaxiques entre S4 et la logique intuitionniste propositionnelle (LJ), formalisés par les traductions réciproques de Gödel [Göd86] et de Goré et Thomson [GT19]. PGeon disposant déjà d'une modélisation intuitionniste, le choix de S4 pose les bases théoriques pour explorer des passerelles de traduction automatiques au sein même de l'outil.

4

Modélisation et évaluation

4.1 Calcul des séquents

La première approche que nous avons prise est celle d'utiliser un calcul des séquents pour S4. Les raisons étaient simples : c'était un système avec lequel mes encadrants et moi-même étions à l'aise, et de plus, il existait un calcul intégrant la sémantique de Kripke [Neg05] mais surtout un uniquement syntaxique et assez récent [Gol25].

Le calcul des séquents est une méthode de déduction syntaxique qui permet de démontrer la validité de formules. Cela consiste en un arbre de preuve ou chaque nœud est la conjonction d'une suite d'hypothèses, suivi du thèse ($\vdash$), et enfin de la disjonction d'une suite de conclusions. C'est-à-dire que chaque nœud à la forme $H_1 \wedge \ldots \wedge H_i \vdash C_1 \vee \ldots \vee C_j$.

L'ensemble de règles peut différer selon la logique choisie (classique, intuitionniste, modale, etc), ainsi il nous fallait trouver l'ensemble de règles syntaxiques pour la logique modale classique S4. Nous avons donc commencé en nous appuyant sur le système uniquement syntaxique de Rea Golan [Gol25] qui propose justement un calcul des séquents propositionnels pour la logique modale S4. Cela convenait très bien, puisque même si ce n'était que propositionnel, cela permet de commencer à travailler sur la partie décidable de la logique modale classique, sans s'encombrer des quantificateurs pour le moment. Voici donc le système de séquents sur lequel nous nous basions.

$$\frac{}{\Gamma, A \vdash A, \Delta}\, ax \qquad \frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}\, cut \qquad \frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} \vee_L$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \wedge_L \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta} \wedge_R \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta} \vee_R$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta} \neg_L \quad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta} \neg_R \quad \frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta} \Rightarrow_L \quad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta} \Rightarrow_R$$

FIGURE 4.1 : Calcul des séquents pour la LK propositionnel

$$\frac{\begin{array}{c}{}^{1:}[\vdash A]\\ \vdots\\ \Gamma \vdash \Delta\end{array}}{\Gamma, \Box A \vdash \Delta}\,\Box_L\ {}^{[1]} \qquad \frac{\Gamma \vdash A, \Delta}{\Gamma \vdash \Box A, \Delta}\,\Box_R \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma, \Diamond A \vdash \Delta}\,\Diamond_L \qquad \frac{\begin{array}{c}{}^{1:}[A \vdash]\\ \vdots\\ \Gamma \vdash \Delta\end{array}}{\Gamma \vdash \Diamond A, \Delta}\,\Diamond_R\ {}^{[1]}$$

FIGURE 4.2 : Calcul des séquents propositionnels pour les modalités

Les règles modales $\Box_L$ et $\Diamond_R$ impliquent d'ajouter des formules dans une liste ad-hoc, pour une réintroduction future avec *cut*. Ces règles énoncées, voici une preuve complète, qui sera expliquée, afin de comprendre le concept. Prenons ici une logique épistémique[1] (souvent basée sur S4), où $\Box A$ signifie que je sais que A, et $\Diamond A$ je crois que A. Ainsi la formule $\Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box\Box B$ signifie : "Si je sais que A, et que A implique B. Alors je sais que B, et je sais que je sais B". Ainsi en sachant A et $A \Rightarrow B$, non seulement je sais que B, mais je sais aussi que je sais B. La preuve se fait également avec les séquents :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash B, A}\,ax \quad \dfrac{}{A, B \vdash B}\,ax}{A, A \Rightarrow B \vdash B}\,\Rightarrow_L \quad {}^{2:}[\vdash A \Rightarrow B]}{A \vdash B}\,cut \quad {}^{1:}[\vdash A]}{\vdash B}\,cut}{\Box(A \Rightarrow B) \vdash B}\,\Box_L \quad {}^{2:}[\vdash A \Rightarrow B]}{\Box A, \Box(A \Rightarrow B) \vdash B}\,\Box_L \quad {}^{1:}[\vdash A]}{\Box A, \Box(A \Rightarrow B) \vdash \Box\Box B}\,\Box_R \times 2}{\vdash \Box A \wedge \Box(A \Rightarrow B) \Rightarrow \Box\Box B}\,\Rightarrow_R; \wedge_L$$

FIGURE 4.3 : Preuve de l'exemple

Outre cet exemple, les logiques modales agrémentées du calcul des séquents sont très utiles pour représenter des systèmes plus complexes. Prenons l'exemple d'un bâtiment, dans lequel il y a un ascenseur, la cage de cet ascenseur, ainsi que des portes à chaque étage. $m :=$ "l'ascenseur est en mouvement", $a :=$ "l'alarme sonne", et $o :=$ "les portes sont ouvertes" (supposons au bon étage, pour la simplicité de l'exemple). Prenons un ensemble simple d'axiomes, ce que nous avons programmé dans cet ascenseur, et fait vérifier par un logiciel d'aide à la preuve : "il est nécessaire que l'ascenseur ne soit pas en mouvement pour que les portes soient ouvertes" ($Ax_1 : \Box(o \Rightarrow \neg m)$) et "il est nécessaire que l'alarme sonne si les portes sont ouvertes et l'ascenseur est en mouvement" ($Ax_2 : \Box(m \wedge o \Rightarrow a)$). Le propriétaire de l'immeuble, soucieux des problèmes judiciaires, veut que si les portes sont ouvertes, il est impossible d'être en mouvement sans déclencher l'alarme. Son but est donc de prouver $Ax_1, Ax_2, o \vdash \Box\neg(m \wedge \neg a)$:

[1]La logique épistémique est une logique modale qui permet de raisonner à propos de la connaissance d'un ou plusieurs agents. Elle permet aussi de raisonner sur les connaissances des connaissances des autres agents

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Ax_1, o, m \vdash a, m}\,ax \quad \dfrac{}{Ax_1, o, m \vdash a, o}\,ax}{Ax_1, o, m \vdash a, m \wedge o}\,\wedge_R \quad \dfrac{}{Ax_1, a, o, m \vdash a}\,ax}{Ax_1, (m \wedge o \Rightarrow a), o, m \vdash a}\,\Rightarrow_L}{Ax_1, \Box(m \wedge o \Rightarrow a), o, m \vdash a}\,\Box_L}{Ax_1, Ax_2, o, m \vdash a}\,Ax_2}{Ax_1, Ax_2, o, m, \neg a \vdash}\,\neg_L}{Ax_1, Ax_2, o, (m \wedge \neg a) \vdash}\,\wedge_L}{Ax_1, Ax_2, o \vdash \neg(m \wedge \neg a)}\,\Rightarrow_R}{Ax_1, Ax_2, o \vdash \Box\neg(m \wedge \neg a)}\,\Box_R$$

FIGURE 4.4 : Preuve de la propriété

Ce système nous a permis de commencer une base pour les fichiers de déclaration en PGeon. Et bien que PGeon utilise la méthode des tableaux, cela était facilement contournable avec des opérateurs *left()* et *right()*, ainsi la règle ax devenait par exemple $left(f)\,;right(f)$ ==X.

Cependant, malgré le fait que nous ayons pu utiliser cette base théorique pour des premières implémentations, il subsistait un certain nombre de problèmes concernant l'implémentation. Tout d'abord, la coupure *cut* n'était que *semi*-éliminable, ce qui rendait la recherche de preuve semi-décidable. De plus, la liste ad-hoc de formules à potentiellement réintroduire était également complexe à gérer, puisque non prévue dans PGeon, et donc nous la faisions dans la liste des règles, ce qui était pénible et difficile à maintenir avec la syntaxe des règles déjà transformées avec des séquents pour les tableaux. Cela faisait déjà quelques contraintes techniques, en plus du fait du peu de littérature pour les calculs des séquents modaux, sans sémantique.

4.2 Tableaux pour S4

C'est pour ces raisons que nous sommes repartis sur l'idée d'utiliser la méthode des tableaux, qui nous permettrait passer outre certains problèmes du calcul des séquents présenté. Il a fallu trouver dans la littérature des systèmes tableaux pour la logique modale, et plus précisément S4. Deux ressources étaient particulièrement utiles, les travaux de M. Fitting [Fit85] et ceux de R. Goré [Gor99]. Le premier proposait une description de la strategie de recherche de preuve, et le second un système théorique de tableaux pour S4. Voici le système que Goré a mis au point, à ajouter aux règles 2.1.

$$\text{T}\ \frac{X\ ;\ \Box P}{X\ ;\ \Box P\ ;\ P} \qquad\qquad \text{S4}\ \frac{\Box X\ ;\ \neg\Box P}{\Box X\ ;\ \neg P}$$

FIGURE 4.5 : Règles T et S4

L'intuition derrière les règles est assez simple. Tout d'abord T, l'axiome énonce $P \Rightarrow \Diamond P$, or $P \Rightarrow \Diamond P \equiv \neg \Diamond P \Rightarrow \neg P \equiv \Box\neg P \Rightarrow \neg P \equiv \Box P \Rightarrow P$, et ainsi $\Box P \Rightarrow P$. Ce que cela signifie, c'est que s'il est nécessaire que P, alors P est vraie dans ce monde

également. L'axiome *S4* suggère $\Box P \Rightarrow \Box\Box P$. Cela représente, comme énoncé plus tôt, la transitivité, et ainsi nous permet d'éliminer la modalité de $\neg\Box P$, pour obtenir $\neg P$. Or par transformation $\neg\Box P \equiv \Diamond\neg P$. La sémantique est donc la suivante, éliminer le diamant de $\Diamond\neg P$ pour "aller" dans le monde où $\neg P$ existe. Dans cette règle $\Box X$ représente l'ensemble des formules f' de la forme $\Box f$, on peut voir que lors de la réécriture de la branche, ces formules ne sont pas supprimées ou réécrites. Cela s'explique par la nature de $\Box$: la nécessité, ce qui est boxé est vrai dans tous les mondes accessibles par $\Vdash$, et donc le monde particulier où l'on a $\neg P$ a aussi $\Box X$, et le $\neg P$ nous donne donc un élément supplémentaire pour le raisonnement. L'autre symbole Z représente toutes les formules qui ne sont ni $\Box X$, ni $\neg\Box P$. En effet, par le fait qu'une formule sans modalité ne se définit que dans le monde local, sans relation aux mondes accessibles, et ne sont donc pas conservées par la réécriture.

L'écriture de preuves tableaux en utilisant une règle comme S4 est assez peu esthétique, puisque montrer des réécritures de branche est plutôt complexe en taille de la preuve. Ainsi, pour cette raison et pour la représentation des preuves tableaux dans des outils informatiques, nous procéderons à des preuves tableaux de tableaux. C'est-à-dire que chaque nœud du tableaux est une branche de tableau.

$$\neg(\Box p \Rightarrow \Box(q \Rightarrow p)) \qquad (\neg\Rightarrow)$$
$$\Box p \ ; \ \neg\Box(q \Rightarrow p) \qquad (S4)$$
$$\Box p \ ; \ \neg(q \Rightarrow p) \qquad (\neg\Rightarrow)$$
$$\Box p \ ; \ q \ ; \ \neg p \qquad (T)$$
$$p \ ; \ q \ ; \ \neg p \qquad (\bot)$$
$$\bot$$

FIGURE 4.6 : Preuve de SYM106

Dans cet exemple, nous prouvons que la formule $\Box p \Rightarrow \Box(q \Rightarrow p)$ [Sid09] est bien un théorème, en obtenant la clotûre de toutes les branches de la négation de la formule. On remarque bien que chaque noeud de la preuve contient un certain ensemble de formules séparées par un point-virgule, ce qui permet de représenter des transformations de branches.

De plus, il faut remarquer que les règles T et $S4$ ont une particularité. En effet, T conserve la formule $\Box P$ (et $S4$ $\Box X$) entre ses prémisses et ses conclusions. C'est indispensable pour pouvoir faire des preuves avec $S4$ après. Ce que cela cache, c'est une application d'une contraction, règle que Goré avait éliminée dans son système, et dont on trouve les traces ici et dans $S4$. Il s'agit de devoir faire une contraction de $\Box P$ avant d'appliquer une règle $T' = \Box P \rightarrow P$, ce qui donne :

$$\dfrac{\dfrac{\dfrac{\Box P}{\Box P \ ; \ \Box P}\text{ Contraction}}{\Box P \ ; \ P}\ T'}$$

Cela permet certes d'éliminer la contraction, mais pourquoi dupliquer $\Box P$? Lors de l'application de $S4$, nous avons besoin de garder nos formules boxées pour les laisser dans les conclusions. Pour la formule $\Box(q \wedge \neg\Box q)$ par exemple, la règle T' ne suffirait pas.

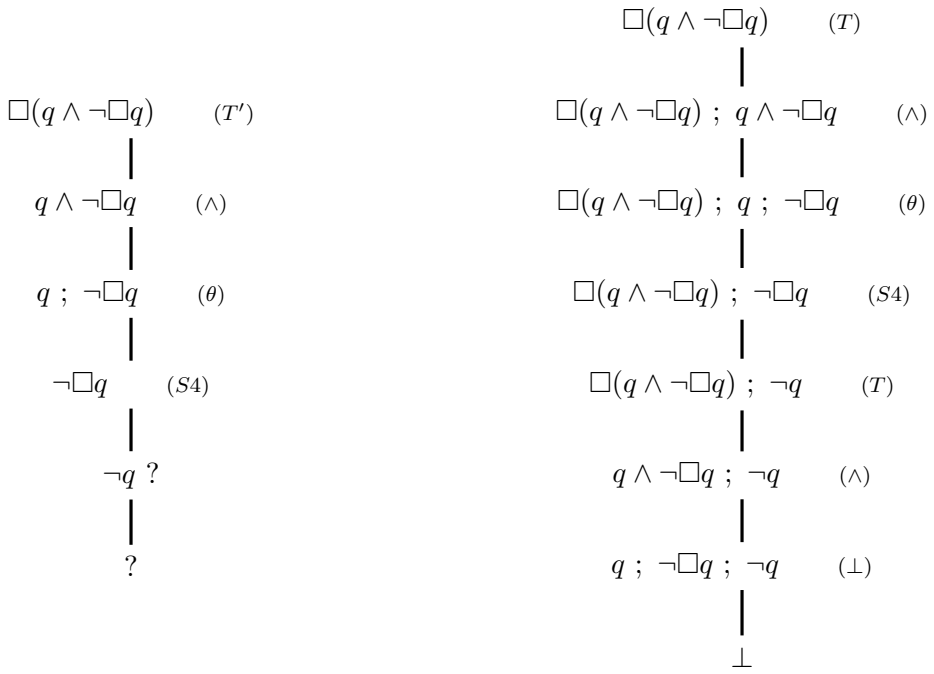


FIGURE 4.7 : Comparaison des comportements des règles T' et T

Et de la même manière, on peut éliminer l'affaiblissement (θ) en le fusionnant avec $S4$, ici l'ensemble Z de formules supprimées :

$$\text{S4}\ \dfrac{\Box X \ ; \ \neg\Box P \ ; \ Z}{\Box X \ ; \ \neg P}$$

FIGURE 4.8 : Intégration de θ dans $S4$

4.3 Implémentation

Cette méthode présentait tout de mêmes des failles pour créer le fichier descriptif de la logique pour *PGeon*, puisqu'il faut garder les formules boxées, sans que T se réapplique dessus en boucle. La première méthode qui vient en tête est l'utilisation d'une liste ad-hoc, mais cette méthode a déjà été écartée, lors des premières implémentations du calcul

des séquents de R. Golan. Nous avons donc contourné le problème en introduisant $\boxdot$, et en changeant les règles T et $S4$ pour pallier ce problème.

$$\text{T}\ \frac{X\ ;\ \Box P}{X\ ;\ \boxdot P\ ;P} \qquad\qquad \text{S4}\ \frac{\boxdot X\ ;\ \neg\Box P\ ;\ Z}{\Box X\ ;\ \neg P}$$

FIGURE 4.9 : Règles $\boxdot$ T et S4

En traitant de cette manière, il n'est plus possible d'appliquer T sur les mêmes formules, et $S4$ rend à nouveau les formule boxées utilisables dans la recherche de preuve. Le choix d'utiliser $\boxdot$ dans $S4$ sera explicité plus bas, lorsque je traiterai plus précisément de la stratégie de recherche de preuve.

Mais la règle la plus difficile à adapter était $S4$. Déjà, cette règle comme énoncée est double contextuelle, une première fois sur $\boxdot X$ pour les formules boxées, et une seconde fois avec Z pour les formules non modales autre que $\neg\Box P$. Réécrire $\neg\Box P$ en $\neg P$ était la partie simple, mais il fallait également faire avec les deux contextes. La première solution que nous avons employée, est de faire une règle $S4$ pour chaque combinaison de $\boxdot X$ et Z possible.

```
1 rule rS4_1x1 : SBox(a()) ; not(box(p())) ; b()  ==>  box(a()) ;
    not(p())
2 rule rS4_1x2 : SBox(a()) ; not(box(p())) ; b() ; c()  ==>  box(a()
    ) ; not(p())
3 ...
4 rule rS4_3x0 : SBox(a()) ; SBox(b()) ; SBox(c()) ; not(box(p()))
    ==>  box(a()) ; box(b()) ; box(c()) ; not(p())
5 ...
6
7 main strategy prove : (rAxiom || alpha || rT || beta || rS4_3x3 ||
    rS4_3x2 || ...)*
```

FIGURE 4.10 : Combinaisons de X et Z pour S4

Cette solution n'était évidemment que temporaire, pour obtenir de premiers résultats sur des problèmes simples. La stratégie était très similaire à la recherche de preuve dans LK propositionnel, avec simplement l'ajout biaisé à gauche des règles $S4$ biaisées à gauche dans un ordre décroissant $\times$ décroissant.

La solution suivante a été d'utiliser les tree-rules de *PGeon* afin d'avoir accès à un premier niveau de contexte pour $S4$. Ces règles se font sur l'ensemble d'une branche, comme dans $S4$, ce qui est parfait. La syntaxe est celle de formules, suivies du reste de la branche ...B et du reste de l'arbre ...T. Avec ces règles, avec l'opérateur de stratégie du biais à gauche, $rS4\_0$ || $rS4\_1$ nous arrivions à un résultat un peu similaire, mais en perdant la complétude de notre système : les formules boxées pouvaient tout à fait matcher les ensembles de formules quelconques (comme A dans 4.11) et se faire supprimer. De plus, cette méthode nécessite l'application directe d'une règle de réécriture pour les $\boxdot$ ($SBox$), puisque les tree-rules ne nous permettent pas de réécrire leur contextes.

```
tree rule rS4_0 : (not(box(P)) ; ...B) | ...T --> (not(P); ...B) |
    ...T
where {
  ...B : SBox(_)
}

tree rule rS4_1 : (not(box(P)) ; A ; ...B) | ...T --> (not(P); ...
   B) | ...T
where {
  ...B : SBox(_)
}
```

FIGURE 4.11 : Tree-rules S4

Cette solution n'était bien sûre pas satisfaisante, alors en attendant plus de développement de *PGeon* sur cette fonctionnalité, nous avons encore modifié le comportement de S4, pour obtenir une version que nous avons gardé un certain temps. Celle-ci utilisait le langage de stratégie pour rechercher les application possibles de $rS4$. L'avantage c'est que cette solution garde les propriétés du système de tableaux proposé par R. Goré, en ré-implémentant l'affaiblissement (qu'il avait éliminé et mis dans $rS4$) et modulo α-réduction.

```
tree rule rS4 : (not(box(P)) ; ...B) | ...T --> (not(P); ...B) |
   ...T
where {
  ...B : box(_)
}

rule rWeak     : X ; Y      ==> X
rule trans_box: SBox(P)    ==> box(P)

strategy modal_jump : (trans_box* &; rWeak*) ; rS4
main strategy prove : (rAxiom || alpha || rT || beta || modal_jump
   )
```

FIGURE 4.12 : Troisième version de S4

$$\text{T}\ \frac{\Box P}{\boxdot P\ ;P} \qquad \text{S4}\ \frac{\Box X, \neg\Box P}{\Box X, \neg P} \qquad \text{trans\_box}\ \frac{\boxdot P}{\Box P} \qquad \text{Weakening}\ \frac{P\ ;\ Q}{P}$$

FIGURE 4.13 : Règles T, S4, trans_box et Weakening, V2

La subtilité était d'utiliser la stratégie pour rechercher les combinaisons d'applications de trans_box et Weakening qui permettaient une application de $rS4$ immédiatement après. Évidemment, cette méthode était coûteuse pour la recherche de preuve, puisque soit Weakening était incontrôlable, et notre système incomplet, soit les recherches de preuves se font en parallèle avec tous les résultats de trans_box $\times$ Weakening possibles.

Cette solution était cependant la meilleure, en attendant que le développeur de *PGeon* travaille sur un moyen d'implémenter $rS4$ comme théorisée plus haut, et a permis de donner des résultats ainsi que des pistes d'amélioration sur la stratégie.

Après plusieurs semaines, nous avons pu développer une solution plus efficace, avec un anti-pattern, c'est-à-dire l'exclusion d'un type de formule dans un pattern. Cet ajout nous permet donc d'avoir une stratégie de recherche de preuve sans la règle d'affaiblissement, ce qui réduit considérablement les coûts en temps et en espace de la recherche de preuve. Ce qui nous donnait donc notre description et stratégie finale pour la logique modale propositionnelle S4. Dans ce fichier descriptif, on retrouve bien les trois parties, la déclaration du type $formula$ et des symboles, les règles de transformation syntaxique, et la stratégie de preuve. On peut remarquer à la ligne 35, que cette version de $rS4$ comprend effectivement un pattern matching sur deux ensembles, B et Z. B matchant avec les formules boxées, et Z avec toutes les autres, sauf $\neg\Box P$.

```
1  main type formula
2
3  function not    : formula -> formula
4  function and    : formula formula -> formula
5  function or     : formula formula -> formula
6  function imp    : formula formula -> formula
7  function equ    : formula formula -> formula
8
9  function box    : formula -> formula
10 function diam   : formula -> formula
11
12 function SBox : formula -> formula
13
14 rule rAxiom : P ; not(P)     ==X
15
16 rule rNNPP  : not(not(P))    ==> P
17 rule rDiam  : diam(P)        ==> not(box(not(P)))
18 rule rNDiam : not(diam(P))     ==> box(not(P))
19
20 rule rAnd : and(P,Q)         ==> P ; Q
21 rule rNOr : not(or(P,Q))     ==> not(P) ; not(Q)
22 rule rNImp  : not(imp(P,Q))    ==> P ; not(Q)
23
24 rule rNAnd  : not(and(P,Q))    ==> not(P) | not(Q)
25 rule rOr    : or(P,Q)            ==> P | Q
26 rule rImp   : imp(P,Q)           ==> not(P) | Q
```

FIGURE 4.14 : Version finale de l'implémentation de S4 (1)

```
rule rEquiv   : equ(P,Q)        ==> imp(P, Q) ; imp(Q, P)
rule rNEquiv  : not(equ(P,Q))    ==> not(imp(P, Q)) | not(imp(Q, P))

rule rT       : box(P)          ==> SBox(P) ; P
rule rWeak    : X ; Y             ==> X
rule trans_box: SBox(P)           ==> box(P)

tree rule rS4 : ( not(box(P)) ; ...Z ; ...B) | ...T --> (not(P);
   ...B) | ...T
where {
  ...B : SBox(_)
  ...Z : ~SBox(_)
}

strategy elim_diam : rDiam || rNDiam
strategy alpha : rNNPP || rAnd || rNOr || rNImp || elim_diam
strategy beta : rNAnd || rOr || rImp || rEquiv || rNEquiv
strategy modal_world : rS4 ; trans_box!

main strategy prove : (rAxiom || alpha || rT! || beta &|
   modal_world)*
```

FIGURE 4.15 : Version finale de l'implémentation de S4 (2)

$$\text{T}\ \frac{\Box P}{\Box P\ ;P} \qquad \text{S4}\ \frac{\boxdot X, \neg\Box P, Z}{\boxdot X, \neg P} \qquad \text{trans\_box}\ \frac{\boxdot P}{\Box P}$$

FIGURE 4.16 : Version finale des règles modales

C'est donc avec cette version de notre calcul que nous avons décidé de faire les benchmarks définitifs.

4.4 Analyseur lexical, syntaxique et traducteur

Afin de pouvoir tester PGeon sur des benchmarks, il devenait nécessaire d'abandonner la rédaction manuelle de problèmes pour se tourner vers les banques de données open source. TPTP [TPT], pour Thousand of Problems for automated Theorem Proving, est une bibliothèque en ligne qui comprend des milliers de problèmes, ainsi que leur statut. Les problèmes sont répartis sur des dizaines de catégories, et peuvent être :

- **Théorème** : Le fichier problème est valide dans une théorie, soit le problème est vrai dans toutes les interprétations,
- **Insatisfiable** : Il n'existe aucune interprétation qui rende le problème vrai,
- **Satisfiable / Non-Théorème** : Il existe une interprétation qui rend le problème vrai,

- **Non-Résolu / Inconnu** : Il n'existe aucune preuve d'une des trois propriétés au dessus pour le problème.

Pour la logique modale, il existe également QMLTP, Quantified Modal Logic for Theorem Proving, qui regroupe 500 problèmes de logique modale, dont une centaine propositionnels et le reste au premier ordre. Le statut des problèmes de QMLTP sont décris sous forme de matrice. Pour notre implémentation de S4, il faut regarder la ligne "S4", colonne "constant", puisque la logique propositionnelle est équivalente à une formule premier ordre sans quantificateur, dont le domaine est donc forcément constant.

Logique	**varying**	**cumulative**	**constant**	**Version**
K	Unsolved	Non-Theorem	Non-Theorem	v1.1
D	Non-Theorem	Non-Theorem	Non-Theorem	v1.1
T	Non-Theorem	Non-Theorem	Non-Theorem	v1.1
S4	Theorem	Theorem	Theorem	v1.1
S5	Theorem	Theorem	Theorem	v1.1

TABLE 4.1 : Statut de SYM144 selon les sémantiques de domaines

Ainsi, nous avons développé un analyseur lexical et syntaxique, ainsi qu'une fonction de traduction, pour pouvoir passer en entrée des problèmes issus des formats TPTP ou QMLTP, et en ressortir des problèmes pour les formats de PGeon ou Tableau Workbench. Cela nous a permis de créer une base de problèmes pour tester ces deux outils, et perfectionner notre implémentation. Cet outil que nous avons nommé TPTPTPTP, pour "TPTP Translation Parser for Tableaux and PGeon" est disponible publiquement ici.

4.5 Résultats

Pour évaluer notre approche, nous avons utilisé un benchmark extrait de la bibliothèque QMLTP, version standardisée de problèmes de logique modale quantifiées [RO12]. En plus, nous avons comparé ces résultats à ceux de l'outil le plus similaire déjà disponible, Tableau Workbench. Le dataset consiste en 100 problèmes de logique modale propositionnelle classique, s'étendant principalement dans le domaine *SYM* (SYntactic Modal), mais certains d'entre eux sont dans *GSY* (Gödel's SYntactic) et *GLC* (Gödel's logic calculi), des ensembles de problèmes dérivés de la traduction de Gödel de la logique intuitioniste dans S4. Les problèmes du dataset sont soit des théorèmes (formules valides) ou des non-théorèmes (satisfiables). Nous avons également sept problèmes dont la résolution est inconnue, que nous avons gardé de côté pour faire plus de tests.

Tout d'abord, la procédure de test a consisté en l'exécution de PGeon et Tableau Workbench avec 240 secondes de timeout sur ces 100 problèmes. Les théorèmes ont été mis sous forme négative, puisque la méthode des tableaux prouve l'insatisfiabilité de formules, et nous attendions donc un arbre clos. Pour les non-théorèmes, les échecs (arbres ouverts) et les timeouts étaient les résultats attendus. Les versions de PGeon, Tableau Workbench et le dataset sont tous disponibles ici. Par ailleurs, nous avions

Attendu	**Problèmes**	**OK**	**KO**	**Succès (%)**
Théorème	73	73	0	100 %
Non-Théorème	27	27	0	100 %
Total	**100**	**100**	**0**	**100 %**

TABLE 4.2 : Résultats de PGeon

trouvé dans le dataset un problème mal étiqueté en non-théorème, au lieu d'insatisfiable. En réétiquetant ce problème, nous obtenons les résultats 4.2.

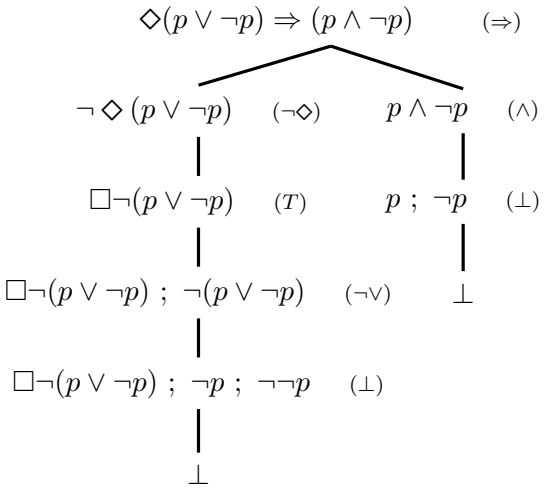


FIGURE 4.17 : Preuve de SYM167+1, mal étiqueté

Les résultats sont très satisfaisants, de plus, les temps d'exécution se sont grandement affaiblis après les retours faits au développeur de PGeon. Nous ne parlons pas des temps exacts dans le benchmark, car ils sont tous inférieurs à 0.001 secondes et considérés comme instantanés.

5

Contribution

Ce sujet de stage portait donc sur une intégration de la logique modale à un outil pratique de génération de prouveurs automatiques. C'est un domaine où comme énoncé précédemment, les logiques modales ne sont que rarement représentées. L'outil le plus proche que nous ayons de PGeon pour la logique modale est Tableau Workbench, qui se base aussi sur les travaux de Raajev Goré. Pour ce qui est des procédures de recherche de preuve, l'article de référence c'est Melvin Fitting qui l'a écrit dans [Fit85]. Mais il utilise un système de formules étiquetées, soient un ajout sémantique dans le moteur syntaxique pour se repérer dans les mondes d'un modèle de Kripke.

Nous nous positionnons donc en tant qu'outil plus fort que Tableau Workbench, car les résultats sont aussi bons, mais PGeon est bien plus expressif, permissif, et permet d'utiliser les logiques quantifiées. C'est justement avec cet aspect orienté preuve automatique en logique modale, où les espaces de recherche de preuves sont souvent énormes dans le peu d'outils déjà existant, et où la flexibilité et l'expressibilité est faible, que nous nous positionnons en solution avec notre travail. Ainsi, nous soumettons également un article scientifique à la conférence internationale *LPAR* (Logic for Programming and Automated Reasoning) à Spetses en Grèce.

De plus, ce stage a permis beaucoup d'améliorations sur l'outil PGeon en lui-même, car avoir un stagiaire qui essaye justement d'implémenter des éléments non prévus originellement, et d'aller aux limites du code. Avec ce stage, parallèlement aux missions initiales, nous avons pu repérer des erreurs dans les opérateurs de stratégie, améliorer les règles portant sur les branches ou encore diviser par 24000 les temps sur certains problèmes. Avoir un utilisateur qui essaye de pousser l'outil sur des zones imprévues a eu un effet bénéfique sur le projet dans son ensemble.

6

Conclusion et ouverture

Ce stage de recherche en laboratoire au LIRMM, dans le domaine de la logique mathématique et la démonstration automatique, était une très bonne expérience. L'objectif principal d'intégrer une logique modale classique propositionnelle dans PGeon a pu être réalisé. Le développement traditionnel de prouveurs automatiques étant fastidieux, en plus que les outils pour la logique modale soient très rares, ce travail sera nous espérons, utile aux chercheurs dans le domaine.

Au cours du stage, nous avons exploré deux approches théoriques différentes, ce qui a permis d'avoir une vision d'ensemble de la littérature sur les méthodes de recherche de preuve en logique modale. Les deux ont soulevé leurs lots de contraintes, et il fallait régulièrement revoir nos directions et hypothèses, ce qui a fait évoluer les outils dont dispose PGeon.

Les résultats obtenus lors des différents benchmarks effectués ont pu confirmer une évolution de nos performances, jusqu'à obtenir une réussite de 100% sur le dataset disponible, avec une exécution presque instantanée. Cela positionne PGeon en position de leader, puisque plus flexible et expressif que les rares solutions déjà existantes. De plus, la stratégie développée permet de vérifier que la procédure de recherche de preuve est terminante, avec du loop-checking. La valorisation scientifique de ces résultats se traduit par la soumission d'un article à la conférence internationale LPAR (*Logic for Programming and Automated Reasoning*) à Spetses, en Grèce.

Bien que l'extension de la logique modale au premier ordre ait été initialement envisagée, l'état de l'art concernant les logiques modales quantifiées (QML) montre que les interactions entre quantificateurs et modalités soulèvent de nombreux problèmes théoriques et sémantiques encore ouverts, tels que le choix entre domaines constants ou variants. Pour le stage, nous avons choisi de stabiliser et obtenir de bons résultats sur un socle propositionnel. Pendant le mois de stage qu'il reste encore, nous essayerons de poser des bases pour de futurs travaux sur l'implémentation de la logique modale S4 classique quantifiée sous domaine constant, soit la version au premier ordre de notre implémentation actuelle.

Bibliographie

[AG09] Pietro Abate and Rajeev Goré. The tableau workbench. *Electronic Notes in Theoretical Computer Science*, 231 :55–67, 2009. Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).

[BRV01] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[Fit85] Melvin Fitting. Proof methods for modal and intuitionistic logics. *Journal of Symbolic Logic*, 50(3) :855–856, 1985.

[Gar24] James Garson. Modal Logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2024 edition, 2024.

[Göd86] Kurt Gödel. *Kurt Gödel : Collected Works : Volume 1 : Publications 1929-1936*, volume 1. Oxford University Press, Oxford, 1986.

[Gol25] Rea Golan. Simple sequent systems for the modal logics k,d,t, and s4. *Logic Journal of the IGPL*, 33(3) :jzaf013, 05 2025.

[Gor99] Rajeev Goré. Tableau methods for modal and temporal logics. In *Handbook of Tableau Methods*, 1999.

[GT19] Rajeev Goré and Jimmy Thomson. A correct polynomial translation of s4 into intuitionistic logic. *The Journal of Symbolic Logic*, 84(2) :439–451, 2019.

[Neg05] Sara Negri. Proof analysis in modal logic. *Journal of Philosophical Logic*, 34(5/6) :507–544, 2005.

[RO12] Thomas Raths and Jens Otten. The QMLTP Problem Library for First-Order Modal Logics. In *Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 454–461. Springer, 2012.

[Sid09] Theodore Sider. *Logic for Philosophy*. Oxford University Press, New York, 2009.

[STK+12] Renate Schmidt, D Tishkovsky, M Khodadadi, P Fontaine, {R A} Schmidt, and S Schulz. Mettel2 : Towards a tableau prover generation platform. In P Fontaine, {R A} Schmidt, and S Schulz, editors, *PAAR-2012 : Proceedings of the Third Workshop on Practical Aspects of Automated Reasoning*, 2012. To appear.

[TPT] The TPTP Problem Library for Automated Theorem Proving. `https://www.tptp.org/`.