45154 DEVARENNE Florian

Formalisation des démonstrations pour l'étude des possibilités informatiques d'assistant de preuves.

En m'intéressant, par le biais de différentes vulgarisations, aux théorèmes d'incomplétude de Gödel j'ai été fasciné par la notion même de complétude et de décidabilité d'une théorie. Ces notions débouchant sur l'existence d'un algorithme démontrant des théorèmes, je me suis demandé la forme que pourrait prendre un tel programme.

La formalisation des démonstrations permet d'effectuer une conversion de la présentation habituelle des démonstrations à une syntaxe formelle précise éliminant toute ambiguïté. Cela est très intéressant dans le contexte de la CPGE où la rédaction des démonstrations fait partie intégrante des compétences évaluées au concours en mathématiques et en informatique.

Positionnement thématique (ÉTAPE 1) :

- *INFORMATIQUE (Informatique Théorique)*
- *MATHEMATIQUES (Autres)*
- *INFORMATIQUE (Informatique pratique)*

Mots-clés (ÉTAPE 1) :

Mots-clés (en français)	**Mots-clés (en anglais)**
déduction naturelle	*natural deduction*
logique	*logic*
assistant de preuve	*proof assitant*
incomplétude	*incompleteness*
décidabilité	*decidability*

Bibliographie commentée

La déduction naturelle est un système formel créé en 1934 par Gentzen dans l'optique de proposer une alternative aux axiomes logiques nécessaires, mais peu naturels, des systèmes à la Hilbert par des règles de déduction/d'inférences. Concrètement il s'agit de présenter les démonstrations sous forme d'arbre ayant pour nœud des règles. La racine étant une règle qui a pour conclusion la formule que l'on souhaite démontrer et les feuilles étant des règles dont les prémisses sont des axiomes. [2]. Un peu ardu à première vue, ce système se montre ensuite presque naturel et traduit pour le moins d'une manière claire les raisonnements que l'on a l'

habitude de faire en mathématiques. L'exemple le plus frappant est la règle d'introduction de l' implication qui traduit le fait de prendre un énoncé pour hypothèse afin d'en démontrer un autre (On veut montrer que A -> B, supposons A, montrons B).

Cette formalisation des démonstrations ouvre de nombreuses possibilités. D' abord, celle d'étudier de plus près ce qu'est une démonstration, de mieux comprendre la structure des raisonnements qui permettent d'établir une vérité implacable, logique, inattaquable. Cette formalisation ouvre aussi la possibilité informatique : ayant une nomenclature claire et une définition de tous les objets constituant les démonstrations on peut à présent les intégrer à un programme informatique. Typiquement un assistant de preuve peut être implémenté. Des logiciels comme Coq,Lean ou encore Isabelle propose différentes formes d' assistanat pour les preuves et notamment une vérification de ces dernières. Les fonctionnalités de ces logiciels peuvent être utile au mathématicien chercheur dans la vérification d'une preuve et dans l'avancé de son travail au quotidien, mais aussi dans l'industrie. En effet, ces logiciels ont gagné en importance ces dernières années grâce à leur utilisation croissante hors de la recherche scientifique théorique. Certains programmes informatiques doivent eux-mêmes être vérifiés par un assistant de preuve, c'est alors souvent la preuve de leur correction et terminaison qui est vérifiée. C'est par exemple le cas dans l'industrie aéronautique pour la vérification d'absence de bug dans les systèmes de pilote automatique mais aussi dans le domaine médical pour les pacemakers ou les robots chirurgicaux. Plus généralement tous les domaines où une erreur logicielle peut entraîner des conséquences sur la vie humaine se tournent vers ces solutions assurant une sureté supplémentaire [3].

Dans un cadre plus théorique, on peut se demander si grâce à un algorithme mettant en place un système de recherche exhaustive il serait possible de chercher une preuve à n' importe quels théorèmes ? Cela est vrai si le système axiomatique que nous utilisons est décidable donc complet (pour toute formule f : on peut soit démontrer f ou son contraire non f) [2]. Les théorèmes d'incomplétude de Godël nous affirment que malheureusement cet espoir est vain : tout système comportant l'arithmétique est incomplet. Cela réduit à néant la perspective d'un prouveur automatique dans les systèmes couramment utilisés comme ZF (Zermelo-Fraenkel). Plus généralement on peut s'intéresser à la logique du première ordre (calcul des prédicats). Cette logique est indécidable, cela signifie qu'il n'existe pas d'algorithme pour déterminer si une formule est valide ou non valide. Ce résultat fut établi par Alonzo Church et Alan Türing en 1936 et 1937. Ils ont aussi montré que cette logique est semi-décidable, c'est-à-dire que nous pouvons écrire un programme, qui prend en entrée une formule et qui décide correctement si la formule est valide ou non lorsqu'il termine (de plus il peut généralement proposer une preuve) et qui termine tout le temps si la formule est valide même si l'exécution peut être très longue [1].

Même s'il est établi qu'aucun algorithme correct terminant tout le temps sera capable de générer des preuves de toutes propositions d'un système incomplet il est intéressant d'étudier comment des preuves peuvent être vérifiées ou encore comment un programme peut assister un humain lors de l'établissement d'une preuve.

Problématique retenue

Comment formaliser et implémenter un moteur de vérification de preuve en déduction naturelle en OCaml, tout en facilitant la compréhension des preuves grâce à une traduction en langage naturel ?

Objectifs du TIPE du candidat

- Formaliser les règles de la déduction naturelle et les implémenter dans un programme Ocaml.

- Mise en place d'un moteur de vérification capable d'accepter ou de rejeter une preuve donnée dans le cadre de la logique des prédicats en Ocaml.

-Mise en place d'un traducteur de preuve en déduction naturelle en Ocaml à des preuves en langage naturel.

- Étudier les possibilités de preuves automatiques et leur intégration dans le traducteur.

- Analyser les performances et les limitations de ces outils.

Références bibliographiques (ÉTAPE 1)

[1] STÉPHANE DEVISMES, PASCAL LAFOURCADE, MICHEL LÉVY : Logique et démonstration automatique : Une introduction à la logique propositionnelle et à la logique du premier ordre *: https://wackb.gricad-pages.univ-grenoble-alpes.fr/inf402/Poly-inf402.pdf*

[2] RENÉ DAVID, KARIM NOUR, CHRISTOPHE RAFFALLI : Introduction à la logique : théorie de la démonstration, 2nd édition *: DUNOD*

[3] CNRS : L'assistant de preuve Coq, un outil essentiel dans la preuve de programmes *: https://www.ins2i.cnrs.fr/fr/cnrsinfo/lassistant-de-preuve-coq-un-outil-essentiel-dans-la-preuve-de-programmes*

[4] PIERRE LE BARBENCHON : Décidabilité de l'arithmétique de Presburger *: leçon agrégation, ENS Rennes*

[5] ALEXANDRE MIQUEL : Les théorèmes d'incomplétude de Gödel *: https://perso.ens-lyon.fr/natacha.portier/enseign/logique/GoedelParAlex.pdf*

[6] BENJAMIN WACK : Cours Benjamin Wack déduction naturelle *: laboratoire Verimag : https://www-verimag.imag.fr/~wack/cours_inf242/node6.html*

DOT

[1] *: Août 2024 : Premier recherche sur les théorèmes d'incomplétudes de Godel et les assistants de preuve*

[2] *: Septembre-Octobre 2024 : Recherche sur la correspondance de Curry-Howard suite aux conseils de mon professeur d'informatique*

[3] *: Octobre-Novembre 2024 : Tentative d'apprentissage du lambda calcul et de compréhension de la correspondance de Curry-Howard*

[4] *: Novembre-Décembre 2024 : Abandon de Curry-Howard, étude d'autre objectif et établissement de la problématique*

[5] *: Février 2025 : Code pour la formalisation de la déduction naturelle en OCaml*

[6] *: Mars 2025-Mai 2025 : Code pour le vérificateur de preuve*

[7] *: Juin 2025 : Code et Test pour assistant/traducteur de preuve*