Leçon 918 : Systèmes formels de preuve en logique du premier ordre. Exemples.

Julie Parreaux

2018 - 2019

Références pour la leçon

[1] Duparc, *La logique pas à pas.*
[2] David, Nour et Raffali, *Introduction à la logique. Théorie de la démonstration.*
[3] Stern, *Fondements mathématiques de l'informatique.*

Développements de la leçon

Complétude de la déduction naturelle — Fonctionnement de Prolog

Plan de la leçon

Introduction **2**

1 Logique du premier ordre **2**
1.1 Syntaxe de la logique du premier ordre [2, p.9] 3
1.2 Sémantique de la logique du premier ordre [2, p.75] 3

2 Systèmes de preuves **3**
2.1 La déduction naturelle 3
2.2 Le calcul des séquents [2, p.185] 4

3 Automatisation des preuves [3, p.231] **5**
3.1 Unification [2, p.248] 5
3.2 Résolution [2, p.264] 5
3.3 Programmer c'est prouver 5

Motivation

Défense

La mathématicien Hilbert au début du $XIX^{\text{ième}}$ a cherché à formaliser et à comprendre la notion de preuve mathématiques. Il cherchait alors le nombre minimal d'axiomes mathématiques nécessaires pour prouver l'ensemble des mathématiques connus. Les logiciens se sont alors intéressés à la notion de preuve et ils ont tentés de répondre à la question : *qu'est-ce qu'une preuve* ?

Les informaticiens se sont ensuite emparés du sujet, notamment avec Curry et Howard dont la correspondance dit que programmer est en réalité prouver (pour cela, ils utilisent le λ-calcul simplement typé et la logique intuitionniste). De cette idée, on a cherché (et on cherche toujours) à automatiser au mieux les preuves à l'aide de la programmation logique ou sous-contrainte, en utilisant des assistants de preuves. Ces derniers sont devenus une aide aux mathématiciens pour certains théorème comme le théorème des quatre couleurs. Mais attention, comme pour tout système informatique, leur correction n'est pas simple à vérifier. Dans ce cas, ils ont tous un noyau qui n'a pas été prouvé : c'est le noyau de confiance de l'assistant auquel on se doit de faire confiance pour assurer la fiabilité des résultats.

Ce qu'en dit le jury

Le jury attend du candidat qu'il présente au moins la déduction naturelle ou un calcul de séquents et qu'il soit capable de développer des preuves dans ce système sur des exemples classiques simples. La présentation des liens entre syntaxe et sémantique, en développant en particulier les questions de correction et complétude, et de l'apport des systèmes de preuves pour l'automatisation des preuves est également attendue.

Le jury appréciera naturellement si des candidats présentent des notions plus élaborées comme la stratégie d'élimination des coupures mais est bien conscient que la maîtrise de leurs subtilités va au-delà du programme.

Introduction

Motivation :

- Hilbert : qu'est-ce qu'une preuve
- Programmer c'est prouver : correspondance de Curry–Howard
- Assistant de preuve

1 Logique du premier ordre

Pourquoi l'étude de la logique du premier ordre ? C'est une logique assez puissante pour exprimer le langage naturel. Il est donc naturel lorsqu'on cherche à formaliser les

preuves mathématiques (qui s'exprime via un langage humain) à se tourner vers cette logique. Cette section est une section de rappel : il n'est pas nécessaire d'en mettre trop (moins il en a, plus on a de systèmes de preuve et autres).

1.1 Syntaxe de la logique du premier ordre [2, p.9]

- *Définition* : Langage de la logique du premier ordre
- *Définition* : Terme et terme clos
- *Définition* : Formule
- *Remarque* : Lien avec le calcul propositionnel
- *Définition* : Variable libre
- *Définition* : Formule close
- *Définition* : Théorie
- *Définition* : Substitution

1.2 Sémantique de la logique du premier ordre [2, p.75]

- *Définition* : Modèle (On définit le modèle au sens de modèle, on se passe de la définition de signature donc on met directement dans la définition de modèle la satisfiabilité.)
- *Remarque* : Pour les théories
- *Exemple* [3, p.220] : Modèle de Herbrand
- *Définition* : Équivalence sémantique
- *Définition* : Théorie contradictoire
- *Problèmes* : Valide et T-Valide

2 Systèmes de preuves

Les systèmes de preuve sont uniquement syntaxique. En effet, une preuve est basée et conduite par la syntaxe de la formule et non sa sémantique. Cependant à l'aide de théorème de complétude et de correction, on arrive à lier syntaxe et sémantique. Nos systèmes de preuve ne font donc pas n'importe quoi.

2.1 La déduction naturelle

La déduction naturelle est un système de preuve dont toutes les déductions se fond sur le but : on passe tout dans le but pour les manipuler. Cela alourdit la manipulation et même si les règles sont relativement facile, il est délicat à automatiser.

Présentation du système de preuve [2, p.24] Nous donnons ici la présentation du système de preuve : les objets qu'il manipule ainsi que les règles de manipulation. On obtient un nombre fini de règles qui permettent de prouver l'ensemble des formules prouvables : ce qui n'était pas évident au départ.

— *Définition* : Séquent

— *Définition* : Règle (voir si on les met en annexe ou non)

— *Définition* : Séquent et formule prouvable par déduction naturelle

Lien entre syntaxe et sémantique [2, p.79] Les systèmes de preuves sont des passerelles entre la syntaxe et la sémantique. Les preuves sont des objets purement syntaxique qui ne fond pas n'importe quoi sur la sémantique. Ce sont l'objets de théorème de correction et de complétude de la logique.

— *Définition* : Théorie consistante, théorie complète

— *Théorème* : Théorie consistante est non-contradictoire (DEV)

— *Corollaire* : Complétude et correction de la déduction naturelle

— *Application* : Théorème de compacité

— *Application* : [2, p.99] Théorème de Lowhein–Skolem

— *Proposition* : Les problèmes Valide et T-Valide sont indécidables

2.2 Le calcul des séquents [2, p.185]

Le calcul des séquents est plus facile à manipuler que la déduction naturelle car on peut manipuler le contexte comme les conclusions (ce qui n'est pas possible en déduction naturelle) : on obtient des règles symétrique. Cela facilite leur manipulation (c'est souvent ce système de preuve qui est automatisé) même s'il est moins naturel que la déduction naturelle pour formaliser des preuves.

— *Définition* : Séquent

— *Définition* : Règle (voir si on les met en annexe ou non)

— *Définition* : Séquent et formule prouvable par calcul des séquents

— *Théorème* : Équivalence des systèmes de preuves

— *Théorème* (ADMIS) : Élimination des coupures (Dans le contexte de la formalisation des mathématiques, la coupure est une règle qui nous permet de faire des lemmes et ainsi de formaliser les jolies preuves. Cependant, dans le cadre de l'automatisation cette règle est ingérable : comment choisir où couper ? Ce théorème implique que nous pouvons nous en passer, ce qui est un premier pas vers l'automatisation des preuves. L'idée de la preuve est de recopier la preuve de la coupure et du lemme partout où on a besoin (formellement c'est une preuve par induction sur l'arbre de preuve).)

3 Automatisation des preuves [3, p.231]

On a ensuite cherché à automatiser les preuve via l'informatique avec l'idée que programmer c'est prouver. Cependant, même si on a un nombre fini de règles, leur automatisation n'est pas si simple (surtout dû au quantificateur existentiel) qui demande l'intervention de l'homme : ce sont les assistants de preuve. On a donc développé plusieurs stratégies basées notamment sur la résolution.

3.1 Unification [2, p.248]

L'unification est un premier pas vers l'automatisation car elle permet de repérer deux termes syntaxiquement équivalents et d'appliquer une substitution afin de les rendre identiques.

- *Définition* : Terme unifiable
- *Définition* : Unificateur (principal)
- *Définition* : Équations unifiables
- *Algorithme* : Unification
- *Théorème* : Correction de l'algorithme d'unification

3.2 Résolution [2, p.264]

La résolution est un système de preuve car il ne possède que deux règles. Il est également très utile dans l'automatisation des preuves.

- *Principe* : On cherche à montrer qu'un ensemble de formule est contradictoire
- *Définition* : Règles (écrire les deux règles et pas une seule)
- *Méthode*
- *Lemme* : Correction de la méthode

3.3 Programmer c'est prouver

La preuve automatique par les programmes est une conséquence de la correspondance de Curry-Howard [1, p.527]. Nous allons étudier quelques exemples de cet adage même si en toute généralité nous devrons parler de λ-calcul simplement typé et logique intuitionniste.

- *Définition* : Clause de Horn
- *Proposition* : Résolution sur les clauses de Horn (DEV)
- *Application* : Prolog et la programmation logique (puis sous contraintes)
- *Exemple* : Requête SQL sur une base de donnée

Références

[1] J. Duparc. *La logique pas à pas*. Presse polytechnicienne et université romandes, 2015.

[2] C. Raffalli, R. David, and K. Nour. *Introduction à la Logique, Théorie de la démonstration (2nd édition)*. Sciences Sup. Dunod, 2004.

[3] J. Stern. *Fondements mathématiques de l'informatique*. Ediscience internationnal, 1990.