Leçon 929 : Le lambda-calcul comme modèle de calcul pur. Exemples.

Julie Parreaux

2018 - 2019

Références pour la leçon

[1] Hankin, *Lambda Calculi : A guide for Computer Scientists.*
[2] Hindley et Seldin, *Lambda Calculus and Combinators : an Introduction.*

Développements de la leçon

μ-récursivité $\Rightarrow$ λ-définissable — Théorèmes de Scott–Curry et Rice

Plan de la leçon

Motivation

Défense

Ce qu'en dit le jury

Il s'agit de présenter un modèle de calcul : le lambda-calcul pur. Il est important de faire le lien avec au moins un autre modèle de calcul, par exemple les machines de Turing ou les fonctions récursives. Néanmoins, la leçon doit traiter des spécificités du lambda-calcul. Ainsi le candidat doit motiver l'intérêt du lambda-calcul pur sur les entiers et pourra aborder la façon dont il permet de définir et d'utiliser des types de données (booléens, couples, listes, arbres).

Introduction

— Introduction d'une logique alternative à la théorie des ensembles par Alonzo Church.
— Modéliser et formaliser les fonctions récursives via le calcul qui est omniprésent.
— Définir la sémantique des langages purement fonctionnels.

1 Le λ-calcul : présentation du modèle

Le λ-calcul est un modèle de calcul dont la syntaxe utilise les λ. On présentera une manière de calculer à l'aide de ce modèle : la β-réduction. Pour finir, on évoque leur lien avec les langages de programmation.

1.1 La syntaxe du λ-calcul

La syntaxe du λ-calcul nous permet de définir l'équivalence via une substitution.

— *Définition* : Langage du λ-calcul
— *Exemple* : Un λ-terme
— *Remarque* : Non-ambiguïté des λ-termes
— *Définition* : Sous-terme sous une partie
— *Définition* : Variable libre et lié
— *Exemple* : Variable libre et lié dans ce terme
— *Définition* : Terme clos
— *Définition* : Longueur d'un terme (nécessaire pour la définition suivante)
— *Définition* : Substitution
— *Définition* : α-conversion (relation d'équivalence définie par induction permettant de rassembler les λ-termes représentant le même calcul)

1.2 La β-réduction

La β-réduction nous permet de calculer grâce aux λ-terme (ce n'est pas la sémantique du λ-calcul en tant que logique mais sa sémantique calculatoire). Elle correspond à une exécution d'un programme.

— *Définition* : β-réduction
— *Définition* : Forme normale
— *Remarque* : On n'a pas toujours existence d'une telle forme
— *Théorème* : Théorème de Church–Rosser
— *Corollaire* : Unicité de la forme normale
— *Définition* : β-équivalence
— *Théorème* : Théorème de Church–Rosser
— *Définition* : Combinateur de point fixe

1.3 Stratégies de réduction et lien avec les langages de programmation

Le λ-calcul est le cœurs de langages de programmation fonctionnel comme LISP ou Haskel. Cependant, une stratégie de réduction est nécessaire et donne le type de langage et ses spécificités.

— *Hypothèse* : La restriction de la β-réduction (expressivité ?)
— *Définition* : λ-abstraction
— *Remarque* : Lien entre λ-abstraction et forme normale
— *Définition* : Réduction par nom
— *Application* : LISP
— *Définition* : Réduction par valeurs
— *Remarque* : Peu efficace sauf mémoïsation
— *Application* : Haskel

2 Un modèle de calcul puissant

Le λ-calcul est un modèle de calcul puissant puisqu'il est aussi expressif que les machines de Turing. De plus, pour pouvoir implémenter des programmes dans les langages de programmation, il nous faut pouvoir implémenter des structures des données grâce à ce modèle de calcul.

2.1 Encoder des données

Le λ-calcul permet d'encoder différentes structures de données nécessaire à la programmation. Il est important de faire passer l'intuition sur ces modèles.

— *Structure de données* : Booléen
— *Application* : Conditionnelle
— *Structure de données* : Paire
— *Structure de données* : Liste
— *Application* : Fonction récursive
— *Structure de données* : Entier de Church (les entiers sont représenter par le travail)
— *Structure de données* : Entier de Barendregt (les entiers sont implémentés pas les pairs)

2.2 Thèse de Church

La thèse de Church est vérifiée par ce modèle de calcul. Nous avons donc une équivalence aux fonctions μ-récursives et aux machines de Turing.

— *Définition* : Fonction λ-définissable
— *Définition* : Fonction μ-récursives
— *Théorème* : Fonction λ-définissable $\Leftrightarrow$ fonction μ-récursive (DEV $\Leftarrow$)
— *Théorème* : Fonction μ-récursives $\Leftrightarrow$ fonction calculable par une machine de Turing
— *Corollaire* : Fonction λ-définissable $\Leftrightarrow$ fonction calculable par une machine de Turing
— *Application* : Thèse de Church

2.3 Théorie de la décidabilité

Grâce à la thèse de Church nous nous ouvrons les portes de la théorie de la décidabilité

— *Définition* : Séparabilité

— *Théorème* : Théorème de Scott–Curry (DEV)

— *Définition* : Fonction non triviale

— *Corollaire* : Théorème de Rice (DEV)

— *Application* : Indécidabilité de l'existence d'une forme normale

Ouverture

La correspondance de Curry–Howard pour le λ-calcul simplement typé.

Références

[1] C. Hankin. *Lambda Calculi : A Guide for Computer Scientists*. Graduate texts in computer science. Clarendon Press, 1994.

[2] J. R. Hindley and J. P. Seldin. *Lambda-Calculus and Combinators : An Introduction*. Cambridge University Press, 2 edition, 2008.