Leçon 912 : Fonctions récursives primitives et non primitives. Exemples.

Julie Parreaux

2018 - 2019

Références pour la leçon

[1] Beauquier, Berstel et Chretienne, *Éléments d'algorithmique.*
[2] Carton, *Langages formels, calculabilité et complexité.*
[3] Lassaigne et Rougemont, *Logique et fondements de l'informatique. Logique du 1er ordre, calculabilité et λ-calcul.*
[4] Wolper, *Introduction à la calculabilité.*

Développements de la leçon

Turing-calculable $\Rightarrow$ μ-récursive — Caractérisation des langages RE

Plan de la leçon

Motivation

Défense

L'historique de ce modèle est moins claire que pour les machines de Turing. En effet, de nombreuses recherches ont été menée dans ce sens pour répondre au dixième problème de

Hilbert (on est loin de l'idée novatrice de Turing). Godel est le premier à présenté les travaux sur les fonctions μ-récursive en 1934 (ils les appellent les fonctions récursives généralisées). Ces travaux se bases sur les fonctions primitives récursives (souvent associée à Kleene) dont les prémisses sont introduits dès les années 1920 et sur les travaux d'Ackermann en particulier qui pointent les limites de celles-ci. Le modèle apparaît donc juste avant les machines de Turing mais L'apparition des trois modèles de calculs classiques sont très proche car produit pendant les début de la recherche sur la théorie de la calculabilité.

Modélisation des fonctions récursives : les fonctions récursives sont un modèles proches des fonctions récursives classique (syntaxe proche du langages des fonctions récursives que l'on manipule). Le deuxième modèle (équivalent) qui modélise les fonctions récursives est le λ-calcul. Ce modèle se concentre sur la sémantique des fonctions récursives et sont à l'origine des sémantiques des langages de programmation purement fonctionnels (comme LISP) où tout est uniquement fonction.

Ce qu'en dit le jury

Il s'agit de présenter un modèle de calcul : les fonctions récursives. S'il est bien sûr important de faire le lien avec d'autres modèles de calcul, par exemple les machines de Turing, la leçon doit traiter des spécificités de l'approche. Le candidat doit motiver l'intérêt de ces classes de fonctions sur les entiers et pourra aborder la hiérarchie des fonctions récursives primitives. Enfin, la variété des exemples proposés sera appréciée.

Introduction

Pré-requis : machine de Turing + langages acceptés et décidés par une machine de Turing + fonction calculée par une machine de Turing

Définition[3, p.115] : Un modèle de calcul.

1 Les fonctions primitives récursives

1.1 Syntaxe et sémantiques

— *Définition* : Syntaxe des expressions des fonctions primitive récursive.

— *Définition* : Sémantique des expressions des fonctions primitive récursive.

— *Définition* : Fonctions primitives récursives

— *Exemples* : Additions, multiplication [2, p.182], flèche de Knuth, factorielle, signe, Fibonacci [4, p.123], exemples des fonctions arithmétiques que l'on utilise dans le premier développement (modulo, division)

Remarque : dans tout le plan on considère des fonctions d'arité quelconque mais qui dont à valeurs dans $\mathbb{N}$. Cependant dans le développement 1 par exemple nous avons besoin de fonction à valeur dans $\mathbb{N}^k$. Il faut donc soit préciser directement dans le plan qu'on peut facilement étendre cette notion (vue comme tuple de fonction) ou au le dire dans la défense de plan ou dans le développement.

1.2 Les prédicats [4, p.125]

— *Définitions* : Prédicats

— *Exemples* : paire, <
— *Définition* : fonction caractéristique
— *Proposition* : caractérisation des prédicats récursifs via les fonctions caractéristiques
— *Exemples* : égalité exemple important, zéro
— *Application* : Les fonctions à support finies sont primitives récursives
— *Définition* : Minimisation bornée Peut être vue comme un prédicat [4, p.128]
— *Proposition* : La minimisation d'une fonction primitive récursive est aussi primitive récursive [4, p.128]

1.3 Les limites des fonctions primitives récursives

— *Théorème* : Il existe une fonction calculable par une machine de Turing qui n'est pas primitive récursive [4, p.129] Attention, ici on n'a pas encore définit ce qu'est la calculabilité (elle vient avec la définition de la thèse de Church) donc la notion de fonction calculable par une machine de Turing est importante.
Bel argument pour montrer que les fonctions primitives récursives ne permettent pas de délimiter l'ensemble des fonctions calculables. On donne ensuit un exemple d'une fonction calculable qui n'est pas primitive récursive
— *Définition* : Ackermann [2, p.183]
— *Proposition* : Ackermann n'est pas primitive récursive[2, p.183]
— *Application* : Les fonctions primitives récursives ne permettent pas de décrire les fonctions calculables.

Lien avec les langages de programmation : Les fonctions primitives récursives sont aussi expressives que les langages ne contenant qu'une boucle FOR.

2 Les fonctions μ-récursive

— *Définition* : Minimisation non bornée [4, p.131]
— *Définition* : Fonction μ-récursive partielle [4, p.131]
— *Définition* : Prédicats sûres [4, p.131]
— *Définition* : Fonction μ-récursive totale [4, p.131]
— *Exemples* : Ackermann [2, p.183]
— *Proposition* : L'inverse d'une μ-récursive est μ-récursive $f^{-1}(k) = \mu.i(f(i) = k)$
— *Exemple* : Inverse de la fonction d'Ackermann
— *Définition* : Fonction μ-récursive partielle [4, p.136]
— *Exemple* : division par 2 ssi le nombre est pair
— *Lien avec les langages de programmation* La minimisation non-bornée introduit la boucle WHILE dans notre modèle de calcul

3 Lien avec la calculabilité

3.1 Lien avec la thèse de Church

Thèse : Thèse de Church pour les fonction μ-récursive

Étude des MT

— *Théorème* : Les fonctions μ-récursives sont exactement les fonctions Turing-calculable. (DEV : Turing $\Rightarrow$ μ-récursive)

— *Corollaire* : Une fonction partielle est μ-récursive ssi elle est Turing-calculable.

— *Application* : La thèse de Church

Étude du λ-calcul [3, p.185]

— *Définition* : Termes.

— *Exemple* : Codage des entiers avec des termes du λ-calcul.

— *Exemple* : La fonction addition par λ-calcul

— *Théorème* : Les fonctions μ-récursives sont exactement représentable par les termes du λ-calcul.

— *Corollaire* : Les fonctions Turing calculable sont exactement représentable par les termes du λ-calcul.

3.2 Les langages récursivement énumérables

— *Définition* : Les langages R et RE + les langages décidables Donner "l'équivalence" entre langages et programmes

— *Problème* : ARRÊT

entrée une machine de Turing déterministe M ; un mot w

sortie oui si $M(w)$ s'arrête ; non sinon

— *Théorème* : Le problème de l'arrêt est indécidable et dans RE.

— *Définition* : La réduction Dessin.

— *Théorème* : Utilisation des réductions

— *Problème* : PAVAGE DE WANG

entrée une famille finie de tuiles

sortie oui si le jeu de tuiles permet de paver le plan ; non sinon

— *Application* : Le pavage de Wang est indécidable

— *Théorème* : Théorème de Rice

— *Application* : Langage vide pour une MT

— *Application* : Théorème de Rice pour fonctions μ-récursives

— *Théorème* : Caractérisation des langages RE (DEV)

Ouverture

Hiérarchie de Chomsky

Références

[1] D. Beauquier, J. Berstel, and P. Chrétienne. *Eléments d'algorithmique*. Manuels informatiques Masson. Masson, 1992.

[2] O. Carton. *Langages formels, calculabilité et complexité*. Vuibert, 2008.

[3] R. Lassaigne and M. de Rougemont. *Logique et fondements de l'informatique. Logique du 1er ordre, calculabilité et λ-calcul.* Hermes, 1993.

[4] P. Wolper. *Introduction à la calculabilité*. Dunod, 2006.