Préparation à l'Agrégation - Option D informatique

Leçon 929: λ-calcul pur comme modèle. Exemples.

Frédéric Blanqui (INRIA)

27/10/18

Programme: "15 Calculabilité, décidabilité et complexité.
(d) Lambda-calcul pur comme modèle de calcul : définition, propriétés (confluence), stratégies, expressivité."

Commentaires du 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)."

Pré-requis et liens avec d'autres leçons: Concernant les notions de variables libres ou liées, d'α-congruence et de substitution, voir la leçon 918 "Systèmes formels de preuve en logique du premier ordre". Concernant l'expressivité, voir les leçons 912 "Fonctions récursives primitives et non primitives" et 913 "Machines de Turing".

Notions et résultats importants à connaître:

- Remplacement vs substitution, α-équivalence, β-réécriture.
- Il y a des termes n'ayant pas de forme β-normale, et des termes ayant une forme β-normale mais pouvant être β-réécrit indéfiniment.
- Par contre, le développement d'un ensemble de β-rédex (= expression β-réductible) termine toujours (théorème des développements finis).
- La β-réécriture est localement confluente.
- La β-réécriture est confluente.
- Différentes stratégies de β-réécriture (e.g. appel par nom, par valeur).
- La stratégie de β-réécriture standard est normalisante (théorème de standardisation): si un terme a une forme normale, celle-ci est atteignable par toute stratégie standard.
- Représentations des entiers en λ-calcul (e.g. Church, Scott).
- Représentation d'autres structures de données.
- Le λ-calcul est Turing-complet.

Livres:

- [Bar84] The lambda calculus: its syntax and semantics (2nd edition), H. Barendregt, North-Holland, 1984. ISBN 0-444-87508-5.
- [Kri90] Lambda-calcul, types et modèles, J.-L. Krivine, Masson, 1990. ISBN 2-225-82091-0. Traduction en anglais disponible sur http://cel.archives-ouvertes.fr/cel-00574575/.
- [Dow10] Les démonstrations et les algorithmes, introduction à la logique et à la calculabilité, G. Dowek, Les éditions de l'École Polytechnique, 2010. ISBN 978-2-7302-1569-5.
- Logique et fondements de l'informatique, logique du 1er ordre, calculabilité et λ-calcul, R. Lassaigne et M. de Rougemont, Hermes, 1993. ISBN 2-86601-380-8.
- [TeR03] Term rewriting systems, Terese, Cambridge tracts in theoretical computer science 55, Cambridge, 2003. ISBN 9780521391153.
- [CF58] Combinatory Logic, H. B. Curry and R. Feys, North-Holland, 1958.

Notes de cours ou transparents:
Il existe de nombreuses notes de cours ou transparents disponibles en ligne par notamment Jean-Jacques Lévy, Jean Goubault-Larrecq, Chantal Berline ou Thérèse Hardin.

- Cours de Jean-Jacques Lévy:
  http://moscova.inria.fr/~levy/courses/X/M1/lambda/
  - transparents d'un cours d'introduction au λ-calcul:
    http://pauillac.inria.fr/~levy/courses/tsinghua/lambda/
  - transparents d'un cours "Reductions and Causality":
    http://pauillac.inria.fr/~levy/courses/tsinghua/reductions/
  - transparents d'un cours "Lambda-calcul et programmation":
    http://moscova.inria.fr/~levy/talks/09college/lbdprog.pdf
- Cours de Jean Goubault-Larrecq:
  http://www.lsv.fr/~goubault/Lambda/lambda.pdf
- Cours de Chantal Berline:
  https://www.irif.fr/~berline/Cours.html
- Cours de Thérèse Hardin:
  http://moscova.inria.fr/~levy/courses/X/M1/lambda/dea-spp/hardin.pdf

Histoire du λ-calcul

Livre: [Bar84, CF58]. Article: [Bar97].

Le λ-calcul a été introduit par Alonzo Church (University of California, Los Angeles) en 1932 comme élément d'un système logique devant permettre la formalisation des mathématiques [Chu32, Chu41]. Le λ-calcul permet de représenter la notion de

fonction au sens calculatoire, contrairement à la notion de fonction utilisée en théorie des ensembles.

Kleene et Rosser, deux étudiants de Church ont cependant montré en 1935 que ce système était en fait logiquement incohérent (de manière similaire au fait que la théorie des ensembles de Frege l'était). Le λ-calcul a cependant eu une postérité très importante. Kleene a en effet montré en 1936 que le λ-calcul permettait de décrire toutes les fonctions récursives partielles, et Turing que ses machines étaient équivalentes au λ-calcul.

Le λ-calcul a ensuite été utilisé comme outil de définition et d'étude des langages de programmation. Il est en particulier à la base du développement des langages de programmation dits "fonctionnels" car permettant d'écrire des programmes qui prennent en argument d'autres programmes et retournent des programmes (LISP, Scheme, OCaml, Haskell, etc.).

Concernant l'extension du λ-calcul en un système logique cohérent capable de représenter les mathématiques, cela a été réalisé plus tard en restreignant la formation des λ-termes par une discipline de typage.

Pour la petite histoire, voici d'où vient le terme de "λ-calcul" [Bar97]. Dans [WR11], Whitehead et Russel utilisaient la notation $2\hat{x} + 1$ pour représenter la fonction qui à x associe $2x + 1$. Church a voulu utiliser la notation $\hat{x}.2x + 1$. L'imprimeur ne pouvant pas mettre $\hat{}$ sur x, il a écrit $\wedge x.2x+1$ à la place. Plus tard, un autre imprimeur a remplacé $\wedge$ par λ.

Bases du λ-calcul

Livres: [Bar84, Kri90].

Les termes ou λ-calcul ou λ-termes sont les termes définis par la grammaire suivante: $t = x|tt|\lambda x.t$ où x appartient à un ensemble infini $\mathcal{V}$ de variables.

Cependant, contrairement aux termes algébriques (du premier ordre), dans le λ-calcul, la substitution d'une variable libre par un terme est une opération non triviale, qui nécessite le renommage des variables liées. Par exemple, $(\lambda x.y)\{y = x\} = \lambda x'.x$. Deux λ-termes équivalents modulo permutation de leurs variables liées sont dits α-équivalents. Dans le λ-calcul, les termes sont considérés modulo α-équivalence. Par exemple, la fonction qui à x associe $2x + 1$ est considérée comme égale à la fonction qui à y associe $2y + 1$: $\lambda x.2x + 1 =_\alpha \lambda y.2y + 1$. Du coup, on doit généralement vérifier que les fonctions, prédicats et relations définis sur les λ-termes sont invariants par α-équivalence.

Il est possible de représenter les λ-termes par des termes algébriques traditionnels du premier ordre de façon à ce que deux λ-termes α-équivalents soient représentés par le même terme. Par exemple, dans les termes de de Bruijn [dB72], une variable x est remplacé par le nombre de λ's qu'il y a entre celle-ci et le λx avec lequel x est lié. Par exemple, $\lambda xy.xy$ est représenté par $\lambda\lambda 21$. Décrire l'effet d'une substitution devient alors non trivial.

La β-réduction $\rightarrow_\beta$ est l'opération consistant à remplacer dans le corps d'une fonction les arguments formels par leurs valeurs. Plus formellement, $\rightarrow_\beta$ est la plus petite relation monotone invariante par α-équivalence contenant les paires $((\lambda x.t)u, t\{x = u\})$.

Confluence et propriété de Church-Rosser

Livres: [Bar84, Kri90].

Dans la suite, on note par $\rightarrow$ la relation de β-réécriture.

La relation $\rightarrow$ est confluente si $\leftarrow^*\rightarrow^* \subseteq \rightarrow^*\leftarrow^*$. Elle a la propriété de Church-Rosser si $\leftrightarrow^* \subseteq \rightarrow^*\leftarrow^*$. Ces deux propriétés sont équivalentes.

Remarque: on ne peut pas montrer la confluence de $\rightarrow$ à partir de la confluence locale (Lemme de Newman) car $\rightarrow$ ne termine pas.

Il y a plusieurs manières de montrer la confluence:

Par exemple, en trouvant une relation confluente R telle que $\rightarrow \subseteq R \subseteq \rightarrow^*$. Méthode de Martin-Löf: prendre pour R la relation $\twoheadrightarrow$ consistant à réduire un nombre quelconque (éventuellement nul) de rédex (éventuellement imbriqués) simultanément.

Variante: réduire tous les réduits d'un coup. Etant donné un terme t, soit t^* le terme obtenu en réduisant tous les rédex de t d'un coup de bas en haut. On a $t \twoheadrightarrow t^*$ et, pour tout u tel que $t \twoheadrightarrow u$ nous avons $u \twoheadrightarrow t^*$.

Remarque: cela marche car, dans le λ-calcul, à chaque position, il y a au plus un rédex et, étant donnés deux rédex, soit ils sont disjoints, soit l'un est inclus dans l'autre sans se superposer. On dit que $\rightarrow_\beta$ est une relation de réécriture orthogonale (linéaire à gauche et sans paire critique), et tout système de réécriture orthogonal est confluent.

Théorème des développements finis

Livres: [Bar84, TeR03].

Cela est un raffinement des propriétés de $\twoheadrightarrow$ où on précise quels rédex doivent être contractés.

Qu'est-ce que le développement d'un ensemble R de rédex d'un terme t? Une R-réduction (ou développement) de t est une suite de contractions de descendants d'un rédex de R. Le théorème des développements finis dit que la R-réduction termine et conflue. De plus, les descendants d'un rédex par deux R-réductions maximales (on dit aussi complètes) distinctes sont identiques.

Il y a plusieurs manières de montrer ce théorème.

Pour éviter de parler de descendants, on étend le λ-calcul avec une nouvelle construction $\texttt{let}\ x = t\ \texttt{in}\ u$ permettant de marquer les rédex de R. On montre ensuite que la relation $\rightarrow'$ telle que $\texttt{let}\ x = t\ \texttt{in}\ u \rightarrow' u\{x = t\}$ (réduction d'un rédex marqué) termine par une méthode à la Tait-Girard en montrant une propriété plus forte, à savoir que $t\sigma$ termine si chaque $x\sigma$ termine, par induction sur t (voir notes de cours de Jean Goubault-Larrecq). Pour monter la confluence de $\rightarrow'$, il suffit alors, par le lemme de Newman, de montrer sa confluence locale.

References

[Bar84] H. Barendregt. *The lambda calculus: its syntax and semantics*. North-Holland, 2nd edition, 1984.

[Bar97] H. Barendregt. The Impact of the Lambda Calculus in Logic and Computer Science. *Bulletin of Symbolic Logic*, 3(2):181–215, 1997.

[CF58] H. B. Curry and R. Feys. *Combinatory logic*. North-Holland, 1958.

[Chu32] A. Church. A set of postulates for the foundations of logic. *Annals of Mathematics*, 33(2):346–366, 1932. Corrections in [Chu33].

[Chu33] A. Church. A set of postulates for the foundations of logic (second paper). *Annals of Mathematics*, 34(4):839–864, 1933.

[Chu41] A. Church. *The calculi of lambda conversion*. Princeton University Press, 1941.

[dB72] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

[Dow10] G. Dowek. *Les démonstrations et les algorithmes, introduction á la logique et á la calculabilité*. Les Éditions de l'École Polytechnique, 2010.

[Kri90] J.-L. Krivine. *Lambda-calcul, types et modèles*. Masson, 1990. English translation in [Kri93].

[Kri93] J.-L. Krivine. *Lambda-calculus, types and models*. Series in computers and their applications Computers and their applications. Ellis Horwood, 1993.

[TeR03] TeReSe. *Term rewriting systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[WR11] A. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1911.