λ-calculus

Samuel Mimram
samuel.mimram@lix.polytechnique.fr
http://lambdacat.mimram.fr

November 2, 2020

We recall that *λ-terms* t are of the form x (a variable) or $\lambda x.t$ (an abstraction) or tu (an application). The *β-reduction* is the closure under context of the relation $(\lambda x.t)u \to t[u/x]$, i.e. the relation generated by

$$\frac{}{(\lambda x.t)u \to t[u/x]} \qquad \frac{t \to t'}{\lambda x.t \to \lambda x.t'} \qquad \frac{t \to t'}{tu \to t'u} \qquad \frac{u \to u'}{tu \to tu'}$$

We write $\overset{*}{\to}$ (resp. $\overset{*}{\leftrightarrow}$) for the reflexive and transitive (resp. and symmetric) closure of $\to$.

1 Reduction graphs

The *reduction graph* of a λ-term t is the graph, whose vertices are λ-terms, defined as the smallest graph such that t is a vertex and there is an arrow between two vertices t and t' whenever $t \to t'$.

1. Write the respective reduction graphs of $(\lambda x.xx)(\lambda y.y)z$ and $(\lambda xy.x)((\lambda x.xx)(\lambda xy.xy))$.
2. Can a reduction graph have loops? be infinite? be infinitely branching?

2 Computing in pure λ-calculus

We encode the booleans true and false as the λ-terms

$$\top = \lambda x.\lambda y.x \qquad\qquad \bot = \lambda x.\lambda y.y$$

1. Define a λ-term if encoding conditional branching: we should have

$$\text{if } \top\, t\, u \overset{*}{\to} t \qquad\qquad \text{if } \bot\, t\, u \overset{*}{\to} u$$

2. Define λ-terms encoding conjunction, disjunction and negation of booleans.
3. Define an encoding of pairs of terms in λ-calculus, as well as projections.

The Church encoding of a natural number n in λ-calculus is

$$\lambda f x.\, \underbrace{f(f \dots (f}_{n \text{ times}}\, x))$$

4. Define the interpretation of the successor, addition, multiplication and exponential functions.
5. Define a function which tests whether its argument, a natural number, is 0 or not.
6. Assuming given the predecessor function, define the subtraction function. Can you see how to define the predecessor?

A *fixpoint combinator* is a term Y such that

$$\mathrm{Y}\ t \overset{*}{\leftrightarrow} t\,(\mathrm{Y}\ t)$$

7. Recall Russell's paradox in naive set theory.
8. Encoding a set t as a predicate which indicates whether an element belongs to it, we can write $t\,u$ instead of $u \in t$, and $\lambda x.t$ instead of $\{x \mid t\}$. Assuming given a term $\neg$ for negation, translate Russell's paradox in λ-calculus, and generalize it in order to obtain a fixpoint combinator Y.

9. Given a term t, show that the β-equivalence class of Y t is always infinite.

10. Program the factorial function in OCaml. Modify your implementation in order not to use the `rec` keyword, but you can use the function `fix` defined by

    ```
    let rec fix f = f (fix f)
    ```

    In practice, what happens when you evaluate this definition? Fix `fix`.

11. Assuming given predecessor, define the factorial function in λ-calculus.

12. The Fibonacci sequence $(\phi_n)_{n\in\mathbb{N}}$ is defined by $\phi_0 = 0$, $\phi_1 = 1$ and $\phi_n = \phi_{n-1} + \phi_{n+2}$. Give a naive OCaml implementation of this function. What is (roughly) its complexity? Provide a saner implementation.

13. Implement the predecessor function in OCaml and in λ-calculus.

14. Show that $\Theta = (\lambda xf.f(xxf))(\lambda xf.f(xxf))$ is also a fixpoint combinator (due to Turing). What is the advantage over Y?

3 Termination of the simply typed λ-calculus

We recall the rules of the simply-typed λ-calculus:

$$\frac{}{\Gamma, x : A, \Gamma' \vdash x : A} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

where, in the first rule, we suppose $x \notin \mathrm{dom}(\Gamma')$. We want to show that every typable term t (in an arbitrary context) is *strongly normalizable*, meaning that there is no infinite reduction from t.

1. Can we show the property by induction on the derivation of the typing of t?

In the course of the proof, will need the following *well-founded induction* principle.

2. Suppose given a set X equipped with a binary relation $\to$ which is *well-founded*: there is no infinite sequence of reductions. Suppose given a property P on the elements of X such that, for every $t \in X$, we have

   $$\forall t \in X.\ ((\forall t' \in X.\ t \to t' \Rightarrow P(t')) \Rightarrow P(t))$$

   Show that $\forall t \in X.\ P(t)$ holds. How can we recover recurrence as a particular case of this?

We define $\mathcal{R}(A)$, the *reducible* terms of type A, by induction by

- $\mathcal{R}(A)$, for A atomic, is the set of strongly normalizable terms,
- $\mathcal{R}(A \Rightarrow B)$ is the set of terms t such that $tu \in \mathcal{R}(B)$ for every $u \in \mathcal{R}(A)$.

A term is *neutral* when it is not an abstraction. We are going to show that following conditions hold:

(CR1) if $t \in \mathcal{R}(A)$ then t is strongly normalizable,
(CR2) if $t \in \mathcal{R}(A)$ and $t \to t'$ then $t' \in \mathcal{R}(A)$,
(CR3) if t is neutral and for every t' such that $t \to t'$ we have $t' \in \mathcal{R}(A)$ then $t \in \mathcal{R}(A)$.

3. Show that these conditions imply that a variable x belongs to $\mathcal{R}(A)$ for every type A.

4. Show the conditions (CR1), (CR2) and (CR3) by induction on A.

5. Suppose that $t[u/x] \in \mathcal{R}(B)$ for every $u \in \mathcal{R}(A)$. Show that $\lambda x.t \in \mathcal{R}(A \Rightarrow B)$.

6. Suppose that $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ is derivable. Show that for all $u_1 \in \mathcal{R}(A_1)$, $\ldots$, $u_n \in \mathcal{R}(A_n)$, we have $t[u_1/x_1, \ldots, u_n/x_n] \in \mathcal{R}(A)$.

7. Show that all typable terms are reducible.

8. Show that all typable terms are strongly normalizable.

9. Use this to show that typable terms are confluent.