Thoughts on Church and Böhm

Joachim Breitner

January 22, 2011

For a self-study course on functional languages at the IIT Bombay, I presented lambda calculus and described Church's results about the unsolvabe problem of determining whether a lambda term has a normal form. We discussed the importance of the result, its connection to Gödel's incompleteness theorem and its impact on Hilbert's program. I was asked to note down my findings, thoughts and interpretation, and additionally comment on Böhm's theorem.

Hilbert and Gödel

Let us start with a very quick historical recap, to put things into order. Hilbert initiated his program around 1920. The aim was to find a complete, consistent logic where all of mathematics could be formalized in and a decision procedure, i.e. an algorithm, which would decide for any proposition formalized in that logic whether it is true or false. Severl people started to work on that, usually basing thir logic on the *Principia Mathematica*, a standard book on logic which was first published between 1910 and 1913.

This program experienced its first setback in 1931, when Gödel showed that any sufficiently powerful logic is either incomplete or inconsistent. In other words: Any consistent system makes statements which are true, but cannot be proven within this system (by stating that they cannot be proven).

But there was still hope that the second part of Hilbert's program could be fulfilled: A decision procedure that decides for each statement whether it is true, false or (this has to be added now, due to Gödel) indeciable.[1] This problem is called the Entscheidungsproblem.

Church

This hope was destroyed by Church. His 1935 paper *An unsolvable problem of elementary number theory*[2] introduced lambda calculus, showed that any recursive function is representable in lambda calculus, argues that algorithms can be implemented using such recursive functions of natural numbers,[3] encodes lambda expressions as natural numbers (Gödel encoding), shows that working with

[1] This is my interpretation of Church's motivation, and not backed by historical reseach.

[2] http://www.fdi.ucm.es/profesor/fraguas/CC/church-AnUnsolvableProblemofElementaryNumberTheory.pdf

[3] "If this interpretation or some similar one is not allowed, it is difficult to see how the notion of an algorithm can be given any exact meaning at all."

these number-encoded lambda expressions (e.g. detecting well formed formulas, application, creating church numbers, performing reduction, detecting normal forms) are effectively calculateable. The unsolvable problem promised in the title is the problem of deciding for (the encoding of) a lambda expression whether it has a normal form.

Church then connects this result to Hilbert's Entscheidungsproblem as follows. The logic of *Principia Mathematica* is strong enough to express the fact that two numbers, when taken as lambda expressions, describe equivalent expressions. If they are, the statement can be proven by exhibiting a finite number of transformation from one into the other. If they are not, this statement can not be proven (assuming the *Principia Mathematica* is consistent[4]). If the Entscheidungsproblem were solved, there were an algorithm which can distinguish these two cases. By some transformation, this would also be able to decide whether a lambda term has a normal form or not, in contradiction to Church's main result. This puts an end to this part of Hilbert's program, too.

Böhm

Böhm's theorem, published in 1968, can be formulated as

> If M and N are different[5] lambda terms in $\beta\eta$ normal form, they describe different functions, i.e. there is a context (a lambda expression with a hole) $\mathcal{C}$ such that $\mathcal{C}[M] \twoheadrightarrow x$ and $\mathcal{C}[N] \twoheadrightarrow y$. This implies that adding $M = N$ as an axiom to the lambda calculus renders it inconsistent and all terms would become equal.

For closed lambda expressions M and N the context can be chosen to be of the form $[\,]L_1 \dots L_r$, i.e. a list of arguments to pass to M and N to show that they are not the same function.

This can be proven[6] in a constructive manner, proceeding by some induction over the depth of the lambda expression and the number of symbols therein and, by tracing the proof backwards, the context can be constructed. In other words, there is an effective calculatable methods that, given two lambda expressions M and N in $\beta\eta$ normal form, decides whether they are convertible into each other and provides a proof for its result.

This is very different from the situation of arbitrary lambda expressions, where the equality is, by Church, an undecidable problem.

Could one not simply unfold all definitions in two functions, e.g. in a programming language like Haskell, to basic lambda expressions, convert to normal form and then easily decide their equality? Unfortunately not, and one reason is that once a fixed point combinator such as

$$\mathbf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) \quad \text{or} \quad \Theta = (\lambda xy.y(xxy))(\lambda xy.y(xxy))$$

shows up, it might not be possible to bring the function into a normal form … and by Chruch, we cannot even tell whether this is the case.

[4] Church assumes ω-consistency. A consistent logic is ω-inconsistent if there is an expressible property P of natural numbers where $P(1)$, $P(2)$, … are all individually provable, but $\exists n.\neg P(n)$ is also provable.

[5] always up to variable renaming

[6] http://www.mathematik.uni-muenchen.de/~joachski/boehm.pdf.gz contains a very terse treatment.