Axiomatizing the Quote

Andrew Polonsky

VU University Amsterdam
De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
`andrew@few.vu.nl`

Abstract

We study reflection in the Lambda Calculus from an axiomatic point of view. Specifically, we consider various properties that the quote $\ulcorner\cdot\urcorner$ must satisfy as a function from Λ to Λ. The most important of these is the existence of a definable left inverse: a term $\mathtt{E}$, called the *evaluator* for $\ulcorner\cdot\urcorner$, that satisfies $\mathtt{E}\ulcorner M\urcorner = M$ for all $M \in \Lambda$. Usually the quote $\ulcorner M\urcorner$ encodes the syntax of a given term, and the evaluator proceeds by analyzing the syntax and reifying all constructors by their actual meaning in the calculus. Working in Combinatory Logic, Raymond Smullyan [12] investigated which elements of the syntax must be accessible via the quote in order for an evaluator to exist. He asked three specific questions, to which we provide negative answers. On the positive side, we give a characterization of quotes which possess all of the desired properties, equivalently defined as being equitranslatable with a standard quote. As an application, we show that Scott's coding is not complete in this sense, but can be slightly modified to be such. This results in a minimal definition of a complete quoting for Combinatory Logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Lambda calculus, combinatory logic, quote operator, enumerator

Digital Object Identifier 10.4230/LIPIcs.CSL.2011.458

1 Introduction

1.1 Coding in mathematical logic

Reflection is a powerful phenomenon in mathematical logic. Its most dramatic application was given by Gödel, who used it in the proof of his famous Incompleteness Theorems, destroying Hilbert's formalist program in its original incarnation (one could call the latter *Naïve Formalism.*) Soon after, it was at the heart of the proofs of equivalence between various models of computation that ultimately provided evidence for Church's thesis. Arithmetization of syntax is also the core component of the enumeration theorem, a result used implicitly in virtually every proof of Recursion Theory.

The ability of a computing system to interpret its own syntax also played a significant role in the evolution of functional programming languages. In one of the early reports on the development of Lisp, John McCarthy [8] introduced the so-called Meta-Circular Evaluator: a Lisp form which can execute an arbitrary list as a Lisp form — a "universal Lisp form." Since then, many languages (including Lisp, Prolog, Smalltalk, and others) have been built ground-up using meta-circular implementation. In the reverse direction, some languages have the "quote" command, which represents expressions of the language within some standard datatype. This operation is not referentially transparent, so the presence of an explicit quote operator in a language (e.g. Lisp) means that the language is not purely functional. Nevertheless, computational reflection provides the language with other powerful capabilities, which were extensively investigated by Brian Cantwell Smith in his PhD thesis [11].

The peculiar use of self-reference made Gödel's argument a favorite among philosophers, and inspired a number of publications in popular science, some of which even attribute a certain mystical element to the work of Gödel. For example, in his introduction to *Gödel, Escher, Bach: an Eternal Golden Braid*, Hofstadter writes: "GEB is in essence a long proposal of strange loops as a metaphor for how selfhood originates."[6] Although Hofstadter's allegorical picture cannot be framed as a scientific thesis, it did stimulate popular interest in computational logic.

The questions of Smullyan were brought to our attention by Henk Barendregt. Of course, they are only a sliver in the more global puzzle of understanding reflection as a distinct phenomenon. There is still lacking a general concept, an all-inclusive definition through which the common features of the constructions in Gödel's theorem, computability, number theory (systems of arithmetic), and set theory could be related. Finding such a concept remains a fascinating open problem.

1.2 Coding of lambda terms

Classically, an *enumerator* is a term $\mathtt{E}$ such that every closed lambda term is convertible[1] to $\mathtt{E}\mathtt{c}_n$ for some natural number n, where $\mathtt{c}_n$ denotes the nth Church numeral. The first enumerator for the lambda calculus was constructed by Kleene [7] in the proof that every lambda-definable function is computable — among the first pieces of evidence for the Church–Turing thesis. Together with the proof that every computable function is lambda-definable, this gave an interpretation of lambda-calculus within itself. Kleene's approach used Gödel's arithmetization of syntax, which codes grammar trees of terms as natural numbers. This has the drawback that an evaluator exists only for terms whose free variables come from a finite set which is fixed in advance.

Mogensen [9] found an elegant self-interpreter which, instead of coding variables by numerals, coded them by themselves. The coding therefore allows an evaluator which is uniform on the set of all (open) lambda terms. Mogensen's construction has a different drawback: it lacks a *discriminator* — a term which can test whether or not two quotes code the same term. However, Barendregt [2] did find a discriminator for Mogensen coding which works for all closed terms.

A more significant distinction between Kleene's enumerator and Mogensen's is that Kleene actually emulates variable binding within the quotes. This requires a number of auxiliary functions to deal with alpha-conversion, making definitions rather complicated. In contrast, Mogensen encodes binders by actual "meta-level" lambdas. This technique is known as Higher Order Abstract Syntax [10], and Mogensen's coding is arguably the most canonical application of it.

In 1992, Berarducci and Böhm gave an improvement on Mogensen's coding such that the evaluator $\mathtt{E}$ is a normal form and $\mathtt{E}\ulcorner M\urcorner$ is strongly normalizing whenever M is. They also listed other properties that a coding might satisfy, and reiterated the problem of axiomatizing the quote as an operator. [5] Our proposed solution appears in Corollary 14.

To keep matters simple, we will restrict attention to the coding of *closed* terms, and work in the combinatory version of the lambda calculus with basis $\{\mathtt{K}, \mathtt{S}\}$. This results in no loss of generality, as all closed lambda terms can always be written in this basis. Indeed,

[1] In fact, in the lambda caluclus all enumerators are actually *reducing*: if $\mathtt{E}$ is an enumerator, then $\forall M \in \Lambda^0\ \exists n \in \mathbb{N}$ s.t. $\mathtt{E}\mathtt{c}_n \twoheadrightarrow M$. Richard Statman gave the first proof of this result using computability theory, and Henk Barendregt provided a constructive adaptation, which can be found in the festschrift of Dirk van Dalen [3].

our constructions can be translated into Mogensen coding rather explicitly. Furthermore, as will be evident from the definitions, the choice of basis has no effect on our results.

2 Setup

2.1 Basic concepts

In what follows, we will need the following concepts. For a thorough introduction, see [1].

▶ **Definition 1.** Let $V = \{v_0, v_1, \dots\}$ be an infinite set of variables.

1. The *lambda terms* are given by the grammar

$$\Lambda ::= V \mid \Lambda\Lambda \mid \lambda V \Lambda$$

2. A subterm occurrence of a variable x in the term M is *bound* if it is inside a subterm of the form $(\lambda x N)$. Otherwise, the occurrence is *free*. $\mathsf{FV}(M)$ denotes the set of variables that have a free occurrence in M. If $\mathsf{FV}(M) = \emptyset$, then M is *closed*, and we write $M \in \Lambda^0$.
3. $M[x := N]$ is the lambda term obtained by renaming bound variables of M to be distinct from the free variables of N, and then plugging in the term N for every free occurrence of x in the resulting M.
4. Lambda terms are considered for their relation of *beta-convertibility* — a congruence generated by the axiom

$$(\lambda x M)N =_\beta M[x := N]$$

5. As a matter of notation, we write
   - $M_1 M_2 \dots M_n$ as a shorthand for $(\dots(M_1 M_2)M_3)\dots)M_n$,
   - $\lambda\vec{x}.M$ as a shorthand for $\lambda x_0(\lambda x_1 \dots (\lambda x_l M)\dots)$.
6. The *combinators* are given by the grammar

$$\mathcal{C} ::= \mathtt{K} \mid \mathtt{S} \mid \mathcal{C}\mathcal{C}$$

   The combinator $\mathtt{SKK}$ is abbreviated by the symbol $\mathtt{I}$.
7. The combinators are considered with the congruence generated by equations
   - $\mathtt{K}xy = x$
   - $\mathtt{S}xyz = (xz)(yz)$
8. Lambda terms are translated into combinators via the map $(\cdot)_{CL} : \Lambda \to \mathcal{C}$:
   - $(x)_{CL} = x$
   - $(MN)_{CL} = (M)_{CL}(N)_{CL}$
   - $(\lambda x.M)_{CL} = \lambda^* x[(M)_{CL}]$,

   where $\lambda^* x[\cdot]$ is given by
   - $\lambda^* x[x] = \mathtt{I}$
   - $\lambda^* x[M] = \mathtt{K}M$, if $x \notin \mathsf{FV}(M)$
   - $\lambda^* x[MN] = \mathtt{S}(\lambda^* x[M])(\lambda^* x[N])$

   Note that when M is closed, $(M)_{CL}$ has no variables (i.e., $(M)_{CL} \in \mathcal{C}$).
9. The basic combinators are represented in Λ by the terms

$$\mathtt{I} = \lambda x.x, \quad \mathtt{K} = \lambda xy.x, \quad \mathtt{S} = \lambda xyz.xz(yz)$$

   In addition, we'll employ the following standard abbreviations:
   - $\mathtt{U}_i^n = \lambda x_0 \dots x_n.x_i$
   - $\overline{\mathtt{K}} = \mathtt{U}_1^1 = \lambda xy.y$

- $\mathtt{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
- $\Omega = (\lambda x.xx)(\lambda x.xx)$
- $[M, N] = \lambda x.xMN, \qquad x \notin \mathsf{FV}(MN)$
- $\langle M_1, \dots, M_n\rangle = \lambda x.xM_1 \dots M_n, \qquad x \notin \mathsf{FV}(M_1) \cup \dots \cup \mathsf{FV}(M_n)$

▶ **Remark 2.** We will often mix together lambda terms and combinators, leaving the translation above implicit in notation. Since we work in combinatory logic, this means that all occurrences of λ are to be eliminated via part 8 of the definition above.

In what follows, we will need to have a standard, reference coding with which others can be compared. Any of those mentioned previously would work; our variant uses pairing to represent the syntax trees.

▶ **Definition 3.** The *standard quote* of M is defined inductively as follows. Let $\mathtt{P} \equiv (\lambda xyz.zxy)_{CL}$ be the pairing combinator. Put

- $\underline{M} \equiv \mathtt{SI}(\mathtt{K}M) = \langle M\rangle \qquad$ if $M \in \{\mathtt{K}, \mathtt{S}\}$,
- $\underline{MN} \equiv \mathtt{P}\,\underline{M}\,\underline{N} = [\underline{M}, \underline{N}] \qquad$ for all M, N.

2.2 Axioms for the quote operator

A *coding* $\ulcorner\cdot\urcorner$ is a map from $\mathcal{C}$ into itself. A term $\ulcorner M\urcorner$ is then called *the quote of* M.[2] Since the primary use of coding consists of manipulating the syntax of terms, most of the properties that we investigate will concern existence of combinators relating the structure of a term to that of its quote. Among these, most attention is given to the Constructor and Destructor axioms. Roughly, the former allows one to obtain the quote of a term from the quotes of its subterms. The latter is dual: it breaks up the term into its subterms (with respect to the quote).

▶ **Definition 4.** (Coding Axioms) Let $\ulcorner\cdot\urcorner : \mathcal{C} \to \mathcal{C}$. We say $\ulcorner\cdot\urcorner$ *satisfies axiom* X from among those below if there exists a combinator X with the stated property.

$$\text{CON } (\textit{constructor}) : \quad \begin{cases} \text{A:} & A\ulcorner M\urcorner\ulcorner N\urcorner = \ulcorner MN\urcorner \\ \text{B:} & B\ulcorner M\urcorner = \ulcorner\ulcorner M\urcorner\urcorner \end{cases}$$

$$\text{DES } (\textit{destructor}) : \quad \begin{cases} \text{P:} & P_i\ulcorner M_0M_1\urcorner = \ulcorner M_i\urcorner, \quad i \in \{0,1\} \\ \text{Z:} & Z_b\ulcorner M\urcorner = \begin{cases} \mathtt{K} & M \equiv b \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases} \quad b \in \{\mathtt{K}, \mathtt{S}\} \end{cases}$$

$$\text{CMP } (\textit{complete}) : \quad \begin{cases} \text{U:} & U\ulcorner M\urcorner = \underline{M} \quad (\text{uncoding}) \\ \text{U}^{-1}\text{:} & U^{-1}\underline{M} = \ulcorner M\urcorner \quad (\text{encoding}) \end{cases}$$

$$\text{E } (\textit{evaluator}) : \quad \mathtt{E}\ulcorner M\urcorner = M$$

$$\Delta \ (\textit{discriminator}) : \quad \Delta\ulcorner M\urcorner\ulcorner N\urcorner = \begin{cases} \mathtt{K} & M \equiv N \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

$$\text{MON } (\textit{monic}) : \quad \forall M, N \in \mathcal{C} \quad \ulcorner M\urcorner = \ulcorner N\urcorner \Longrightarrow M \equiv N$$

$$\text{SOL } (\textit{solvable}) : \quad \forall M \in \mathcal{C} \quad \ulcorner M\urcorner \text{ is solvable}$$

[2] Some authors would call $\ulcorner M\urcorner$ a *quasiquote*, but we will not make this distinction here.

▶ **Remark 5.** Smullyan called a coding satisfying CON *admissible*, and a coding satisfying DES *preadmissible* [12, p.367]. He asked whether either implies the other, and whether an evaluator can be constructed from CON. All three questions have negative answers.

▶ **Remark 6.** Axiom B appears to be too strong: if we want to requote M, why should we care about *the particular* $=_\beta$*-representative* of $\ulcorner M \urcorner$? It may be more reasonable to require

$$\mathsf{B}^-: \qquad B^- \ulcorner M \urcorner = \ulcorner N \urcorner, \text{ where } N = \ulcorner M \urcorner$$

Nevertheless, we will proceed with Smullyan's original formulation.

The axioms above are the primary focus of our attention. In studying them, the following auxiliary properties are useful.

▶ **Definition 7.** We introduce two additional axioms

$$\mathrm{Z}_? \text{ (}\textit{leaf test}\text{):} \qquad Z_? \ulcorner M \urcorner = \begin{cases} \mathtt{K} & M \in \{\mathtt{K}, \mathtt{S}\} \\ \overline{\mathtt{K}} & M \equiv M_0 M_1 \end{cases}$$

R_D (*range test*): $\exists$ *c.e.* $D \subseteq \mathcal{C}$, $\mathsf{Range}(\ulcorner \cdot \urcorner) \subseteq D$, $\exists R_D \in \mathcal{C}$, $\forall N \in D$:

$$R_D N = \begin{cases} \mathtt{K} & \exists M.\ N = \ulcorner M \urcorner \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

3 Results

3.1 Elementary properties

▶ **Proposition 8.** *Let* $\ulcorner \cdot \urcorner$ *be a coding. Then the following implications hold:*
1. $\ulcorner \cdot \urcorner \equiv \underline{\cdot} \implies$ CON $\wedge$ DES $\wedge$ E $\wedge$ Δ
2. Z $\implies$ SOL, DES $\implies$ MON, $\Delta \implies$ MON $\wedge$ Z
3. CMP $\implies$ CON $\wedge$ DES $\wedge$ Δ
4. DES $\implies$ U, U $\implies$ E $\wedge$ Δ

Proof. 1. We verify that the standard coding has all of the properties of interest.
- Let $P_i = \langle \mathtt{U}^1_i \rangle_{CL} = \mathtt{SI}(\mathtt{KU}^1_i)$. Then

$$\begin{aligned} P_i[M_0, M_1] &= \mathtt{I}[M_0, M_1](\mathtt{KU}^1_i[M_0, M_1]) \\ &= (\lambda x_0 x_1 . x_i) M_0 M_1 = M_i \end{aligned}$$

In particular, $P_i \underline{M_0 M_1} = \underline{M_i}$.
- Let $Z_? = (\lambda x . x \mathtt{U}^3_2 \mathtt{I})_{CL}$. (From now on, $(-)_{CL}$ will be left implicit as per Remark 2.) Then

$$\begin{aligned} Z_? \langle x \rangle &= \mathtt{K} \\ Z_? [x, y] &= \overline{\mathtt{K}} \end{aligned}$$

In particular, $Z_? \underline{MN} = Z_? [\underline{M}, \underline{N}] = \overline{\mathtt{K}}$, and $Z_? \underline{\mathtt{K}} = Z_? \underline{\mathtt{S}} = \mathtt{K}$.
- With Z$_?$ satisfied, it is trivial to get full Z. Since $\mathtt{K}, \mathtt{S}$ are normal forms, by Böhm's theorem [1], there exist closed terms $\vec{Q}$ such that $\underline{\mathtt{K}} \vec{Q} = \mathtt{K}$ and $\underline{\mathtt{S}} \vec{Q} = \overline{\mathtt{K}}$. Take

$$\begin{aligned} Z_{\mathtt{K}} x &= \textsc{if } Z_? x \textsc{ then } x \vec{Q} \textsc{ else } \overline{\mathtt{K}} \\ Z_{\mathtt{S}} x &= \textsc{if } Z_? x \textsc{ then } x \vec{Q} \overline{\mathtt{K}} \mathtt{K} \textsc{ else } \overline{\mathtt{K}} \end{aligned}$$

- So far we have proved that $\underline{\cdot}$ satisfies DES. To satisfy axiom A, simply take $A = \mathtt{P}$. Furthermore, this is the representative of the β-equivalence class of $\underline{MN}$ that was chosen by Definition 3: $\underline{MN} \equiv (\mathtt{P}\,\underline{M})\,\underline{N}$.
- Using a fixed-point combinator, put

$$Bx = \text{IF } Z_? x \text{ THEN } (Z_{\mathtt{K}} x)\underline{\underline{\mathtt{K}}}\,\underline{\underline{\mathtt{S}}} \text{ ELSE } \mathtt{P}(\mathtt{P}\underline{\mathtt{P}}(B(P_0 x)))(B(P_1 x))$$

- Evaluator is easy for the standard coding:

$$\mathtt{E}x = \text{IF } Z_? x \text{ THEN } x\mathtt{I} \text{ ELSE } x(\lambda xy.\mathtt{E}x(\mathtt{E}y))$$

- So is the discriminator:

$$\begin{aligned}\Delta xy = &\text{IF } Z_? x\\ &\quad \text{THEN IF } Z_{\mathtt{K}} x \text{ THEN } Z_{\mathtt{K}} y \text{ ELSE } Z_{\mathtt{S}} y\\ &\quad \text{ELSE IF } Z_? y \text{ THEN } \overline{\mathtt{K}} \text{ ELSE } (\Delta(P_0 x)(P_0 y))(\Delta(P_1 x)(P_1 y))\overline{\mathtt{K}}\end{aligned}$$

2. That Z $\Longrightarrow$ SOL is an immediate consequence of the Genericity Lemma [1, 14.3.24]: if $ZM = \mathtt{K}$ for unsolvable M, then $Zm = \mathtt{K}$ for all M, contradicting condition Z.
That $\Delta \Longrightarrow$ MON $\wedge$ Z is also immediate.
To see that DES $\Longrightarrow$ MON we proceed by induction on the height of M. The base step is assured by Z. If $M \equiv M_0 M_1$, and $N \equiv N_0 N_1$, then $(M \not\equiv N) \Longrightarrow (M_i \not\equiv N_i)$ for some i. If $\ulcorner M \urcorner = \ulcorner N \urcorner$, then applying the i'th projection contradicts the inductive hypothesis.

3. Use translation to $\underline{\cdot}$ and back.

4. Take

$$Ux = \text{IF } Z_? x \text{ THEN } (Z_{\mathtt{K}} x)\underline{\mathtt{K}}\,\underline{\mathtt{S}} \text{ ELSE } A_s(U(P_0 x))(U(P_1 x))$$

where A_s is a combinator witnessing axiom A for the standard coding.
Then take

$$\mathtt{E} = \mathtt{E}_s \circ U, \qquad \Delta = \Delta_s \circ U$$

where $\mathtt{E}_s$ and Δ_s are the evaluator and the discriminator for the the standard coding which were constructed in part 1. ◀

3.2 Negative results

Notice that when the coding $\ulcorner\cdot\urcorner$ is a constant map, then it satisfies CON but neither Z nor E. Thus, as pointed out by an anonymous referee, two of Smullyan's questions have trivial answers.

A slight modification to the standard coding gives a counterexample that is also monic and solvable.

▶ **Theorem 9.** *There exists a map* $\ulcorner\cdot\urcorner$ *which satisfies* CON, MON, SOL, *and* P, *yet neither* Z *nor* E. *In particular,* CON $\not\Longrightarrow$ DES.

Proof. Define $\ulcorner\cdot\urcorner$ by

- $\ulcorner M \urcorner \equiv [\Omega M]$ if $M \in \{\mathtt{K}, \mathtt{S}\}$
- $\ulcorner MN \urcorner \equiv \mathtt{P}\ulcorner M \urcorner \ulcorner N \urcorner = [\ulcorner M \urcorner, \ulcorner N \urcorner]$

Note that $\ulcorner\cdot\urcorner$ is monic, solvable, and satisfies A, P, and $Z_?$ via the same combinators as the standard coding. The combinator witnessing B must be modified ever so slightly:

$$Bx = \text{IF } Z_? x \text{ THEN } (Z_{\mathtt{K}} x)\ulcorner\ulcorner\mathtt{K}\urcorner\urcorner\ulcorner\ulcorner\mathtt{S}\urcorner\urcorner \text{ ELSE } \mathtt{P}(\mathtt{P}\ulcorner\mathtt{P}\urcorner(B(P_0 x)))(B(P_1 x))$$

To finish the proof, note that if $Z(\lambda x.x(\Omega M)) = \mathtt{K}$, then by Genericity Lemma [1, 14.3.24] we have $Z(\lambda x.x(\Omega M)) = Z(\lambda x.x(\Omega N))$. Therefore, no term can satisfy axiom Z. By the same token, no evaluator can exist, for its value on $\ulcorner\mathtt{K}\urcorner$ would necessarily agree with that on $\ulcorner\mathtt{S}\urcorner$. ◀

▶ **Theorem 10.** *There exists a map* $\ulcorner\cdot\urcorner$ *satisfying* Z *and* P *which does not satisfy* A*. Thus* DES $\not\Rightarrow$ CON*. Furthermore,* $\ulcorner\cdot\urcorner$ *is monic and solvable.*

Proof. For $M \in \mathcal{C}$, let $s(M)$ denote the size of the syntax tree of M, defined inductively by $s(\mathtt{K}) = s(\mathtt{S}) = 1,\ s(MN) = 1 + s(M) + s(N)$. Certainly, $s(M)$ can be easily computed from $\underline{M}$:

$$\tilde{s}x = \text{IF } Z_? x \text{ THEN } \mathtt{c}_1 \text{ ELSE } \mathtt{c}_+\mathtt{c}_1(\mathtt{c}_+(\tilde{s}(P_0 x))(\tilde{s}(P_1 x)))$$

where $\mathtt{c}_n$ is the n'th Church numeral, and $\mathtt{c}_+$ denotes addition.

For $n \in \mathbb{N}$, let H_n be a lambda term encoding the first n values of the characteristic function of the halting problem. Specifically, we put $H_n = \langle h_0, h_1, \ldots, h_{n-1}\rangle = \lambda z.z\vec{h}$, where

$$h_i = \begin{cases} \mathtt{K} & \varphi_i(i)\downarrow \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

For $0 \leq k \leq n$, let Π^n_k be such that $\Pi^n_k\langle M_1, \ldots, M_n\rangle = \langle M_1, \ldots, M_k\rangle$. For example, Π^n_k could be obtained by taking $\Pi^n_k = \Pi\mathtt{c}_n\mathtt{c}_k$, where

$$\Pi nk = \lambda x.\mathtt{B}x(k\mathtt{B}(\lambda s.k\langle\mathtt{I}\rangle(n\mathtt{K}s)))$$

(Here $\mathtt{B}$ is the composition combinator $\lambda xyz.x(yz)$.)

Finally, put

$$\ulcorner M\urcorner = [\underline{M}, H_{s(M)}] \tag{1}$$

Trivially, MON and SOL are satisfied. To see that this coding satisfies axiom Z, we simply compose the combinator Z_b for the standard coding with the first pair-projection:

$$(\lambda x.Z_b(x\mathtt{K}))\ulcorner M\urcorner = \begin{cases} \mathtt{K} & M \equiv b \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases} \qquad b \in \{\mathtt{K}, \mathtt{S}\}$$

For the ith projection, we use the standard combinator P_i with the auxiliary combinators we defined above:

$$\begin{aligned}
&(\lambda x.(\lambda y.[y, \Pi(\tilde{s}(x\mathtt{K}))(\tilde{s}y)(x\overline{\mathtt{K}})])(P_i(x\mathtt{K})))\ulcorner M_0M_1\urcorner \\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i(\ulcorner M_0M_1\urcorner\mathtt{K})) \\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i([\underline{M_0M_1}, H_{s(M_0M_1)}]\mathtt{K})) \\
=&(\lambda y.[y, \Pi(\tilde{s}(\ulcorner M_0M_1\urcorner\mathtt{K}))(\tilde{s}y)(\ulcorner M_0M_1\urcorner\overline{\mathtt{K}})])(P_i\underline{M_0M_1}) \\
=&(\lambda y.[y, \Pi(\tilde{s}\underline{M_0M_1})(\tilde{s}y)(H_{s(M_0M_1)})])\underline{M_i} \\
=&[\underline{M_i}, \Pi(\tilde{s}\underline{M_0M_1})(\tilde{s}\underline{M_i})\langle h_0, \ldots, h_{s(M_0M_1)-1}\rangle] \\
=&[\underline{M_i}, \langle h_0, \ldots, h_{s(M_i)-1}\rangle] \\
=&\ulcorner M_i\urcorner
\end{aligned}$$

Hence (1) satisfies DES. But notice that

$$\varphi_e(e){\downarrow} \iff \ulcorner \mathtt{K}^e \mathtt{I} \urcorner \overline{\mathtt{K}} \mathtt{U}^e_e = \mathtt{K}$$

Therefore, if there was a combinator for axiom A, we could decide the halting problem by checking whether $\mathtt{c}_e(A \ulcorner \mathtt{K} \urcorner) \ulcorner \mathtt{I} \urcorner \overline{\mathtt{K}} \mathtt{U}^e_e$ equals $\mathtt{K}$ or $\overline{\mathtt{K}}$. Such an A cannot exist. Thus $\ulcorner \cdot \urcorner$ does not satisfy CON. ◀

Notice that non-computability of the coding $\ulcorner \cdot \urcorner$ was essential in the proof above. Indeed, if the coding was computable, then axiom U^{-1} would be satisfied. By Proposition 8, part 4, the destructor axiom would make the coding complete. Then by part 3, it would satisfy CON.

3.3 Positive results

The next natural question is what additional property could be sufficient for the equivalence CON $\iff$ DES to hold. It turns out that existence of a discriminator goes quite far in this direction.

▶ **Theorem 11.** $\Delta \wedge \mathrm{U}^{-1} \implies \mathrm{U}$.

Proof. To construct U, we need to uniformly enumerate all combinators built up from $\mathtt{K}$ and $\mathtt{S}$. Recall that $[M_n]$ is a *uniform enumeration* of $\{M_n\}$ if for each k, there is some X_k such that

$$[M_n] = [M_0, [M_1, [M_2, \ldots [M_k, X_k] \ldots]$$

That is, $[M_n]$ is an infinite stream whose elements form the sequence $\{M_n\}$. The following functions operate on streams:

$$\begin{aligned} \mathtt{Map} f m &= [f(m\mathtt{K}), \mathtt{Map} f (m\overline{\mathtt{K}})] \\ \mathtt{Fold} f m &= f(m\mathtt{K})(\mathtt{Fold} f (m\overline{\mathtt{K}})) \\ \mathtt{Merge} m n &= [m\mathtt{K}, [n\mathtt{K}, \mathtt{Merge}(m\overline{\mathtt{K}})(n\overline{\mathtt{K}})]] \end{aligned}$$

(These definitions implicitly make use of fixed-point combinators.)

Now we define the *standard enumeration* of CL terms to be

$$\mathcal{C} = [\mathtt{K}, [\mathtt{S}, \mathtt{Fold}\ \mathtt{Merge}\ (\mathtt{Map}\ (\lambda s.\mathtt{Map}\ s\ \mathcal{C})\ \mathcal{C})]]$$

It is straightforward to verify that for each $M \in \mathcal{C}$ there is a unique n such that $M \equiv C_n$, where $\mathcal{C} = [C_0, [C_1, \ldots]]$. But here we need the combinators to be quoted, hence we define

$$\underline{\mathcal{C}} = [\underline{\mathtt{K}}, [\underline{\mathtt{S}}, \mathtt{Fold}\ \mathtt{Merge}\ (\mathtt{Map}\ (\lambda s.\mathtt{Map}\ (\mathtt{P}s)\ \underline{\mathcal{C}})\ \underline{\mathcal{C}})]]$$

where $\mathtt{P}$ is the pairing combinator from Definition 3. (Note that this notation is overloaded; we don't mean that $\underline{\mathcal{C}}$ is the standard quote of $\mathcal{C}$.)

Define

$$U_0 s x = \text{IF } \Delta x (U^{-1}(s\mathtt{K})) \text{ THEN } s\mathtt{K} \text{ ELSE } U_0(s\overline{\mathtt{K}})x$$

$$U = U_0 \underline{\mathcal{C}}$$

Note that $U_0 \underline{\mathcal{C}} \ulcorner M \urcorner = \underline{M}_n$, where $n = (\mu k)(M \equiv M_k \in \mathcal{C})$. Thus

$$U \ulcorner M \urcorner \equiv \underline{M}.$$

This completes the proof of the theorem. ◀

▶ **Theorem 12.** *Suppose* $\ulcorner\cdot\urcorner$ *is a coding which satisfies* Δ*. Then* $\mathrm{U}^{-1} \iff \mathrm{A}$*. In particular,* $\mathrm{CON} \wedge \Delta \implies \mathrm{DES}$*.*

Proof. ($\Rightarrow$) By the theorem above and Proposition 8.3,
$\Delta \wedge \mathrm{U}^{-1} \implies \mathrm{CMP} \implies \mathrm{CON} \wedge \mathrm{DES} \implies \mathrm{A}$.

($\Leftarrow$) Suppose Δ and A are satisfied. Put

$$\begin{aligned} U^{-1}x = \; & \textsc{if } Z_? x \\ & \textsc{then if } Z_{\mathtt{K}} x \textsc{ then } \ulcorner\mathtt{K}\urcorner \textsc{ else } \ulcorner\mathtt{S}\urcorner \\ & \textsc{else } A(U^{-1}(P_0 x))(U^{-1}(P_1 x)) \end{aligned}$$

By induction, $U^{-1}\underline{M} = \ulcorner M \urcorner$, hence U^{-1} is satisfied. ◀

It remains to consider the question of reconstructing the quote from the Destructor axioms. The problem with using the approach of Theorem 11, where we try to "guess" the quote of a term by comparing every possibility to the input, is that we have no information on the space of these possibilities. This is where the range test comes in. Recall that the statement of this axiom is

$$\exists \text{ c.e. } D \subseteq \mathcal{C},\ \mathsf{Range}(\ulcorner\cdot\urcorner) \subseteq D,\ \exists R_D \in \mathcal{C},\ \forall N \in D :$$
$$R_D N = \begin{cases} \mathtt{K} & \exists M.\ N = \ulcorner M \urcorner \\ \overline{\mathtt{K}} & \text{otherwise} \end{cases}$$

With the axiom above, we state the final theorem.

▶ **Theorem 13.** $\mathrm{DES} \wedge \mathrm{R}_D \implies \mathrm{U}^{-1}$.

Proof. As in the proof of Theorem 11, we construct $U^{-1}x$ by looking at all possibilities until we find one that matches x, according to the standard discriminator. Let D enumerate a superset of $\mathsf{Range}(\ulcorner\cdot\urcorner)$. We receive this fact as a uniform enumeration $D = [M_n]$, such that for each n one has $R_D M_n = \mathtt{K}$ if $M_n = \ulcorner N \urcorner$ and $R_D M_n = \overline{\mathtt{K}}$ otherwise. Furthermore, every quote $\ulcorner N \urcorner$ appears in the list D: $\forall N \exists n\ \ulcorner N \urcorner = M_n$.

As per Proposition 8.4, let U witness axiom U for $\ulcorner\cdot\urcorner$. Now put

$$U^{-1}x = \mathtt{Fold}\ (\lambda ht.\ \textsc{if } (R_D h)(\Delta x (U h))\overline{\mathtt{K}} \textsc{ then } h \textsc{ else } t)\ D$$

It is routine to verify that $U^{-1}\underline{M} = \ulcorner M \urcorner$ for each M. ◀

▶ **Corollary 14.** *For a complete coding, one of the following suffices:*

$$\begin{aligned} &\mathrm{A} \wedge \Delta \\ &\mathrm{U}^{-1} \wedge \Delta \\ &\mathrm{U}^{-1} \wedge \mathrm{DES} \\ &\mathrm{R}_D \wedge \mathrm{DES} \end{aligned}$$

Proof. By the theorems 8 through 13. ◀

4 An application: minimal codings

We conclude by applying the above results to a practical problem concerning minimal codings.

Below we define what is probably the simplest non-trivial coding in Combinatory Logic. According to Barendregt, it was first suggested by Dana Scott in a letter to Troelstra. [4]

▶ **Definition 15.** (Scott's coding) Let

- $\ulcorner b \urcorner = \mathtt{K}b$, $b \in \{\mathtt{K}, \mathtt{S}\}$
- $\ulcorner MN \urcorner = \mathtt{S}\ulcorner M \urcorner \ulcorner N \urcorner$

However, it turns out that Scott's coding is not preadmissible, i.e., does not satisfy DES.

▶ **Proposition 16.** *Scott's coding is not complete.*

Proof. Observe that

$$\begin{aligned}
\ulcorner \Omega \urcorner &= \ulcorner \mathtt{SII(SII)} \urcorner \\
&= \mathtt{S}\ulcorner \mathtt{SII} \urcorner \ulcorner \mathtt{SII} \urcorner \\
&= \lambda z.(\ulcorner \mathtt{SII} \urcorner z)(\ulcorner \mathtt{SII} \urcorner z) \\
&= \lambda z.(\mathtt{S}\ulcorner \mathtt{SI} \urcorner \ulcorner \mathtt{I} \urcorner z)(\mathtt{S}\ulcorner \mathtt{SI} \urcorner \ulcorner \mathtt{I} \urcorner z) \\
&= \lambda z.(\ulcorner \mathtt{SI} \urcorner z(\ulcorner \mathtt{I} \urcorner z))(\ulcorner \mathtt{SI} \urcorner z(\ulcorner \mathtt{I} \urcorner z)) \\
&= \lambda z.(\mathtt{S}\ulcorner \mathtt{S} \urcorner \ulcorner \mathtt{I} \urcorner z(\mathtt{KI}z))(\mathtt{S}\ulcorner \mathtt{S} \urcorner \ulcorner \mathtt{I} \urcorner z(\mathtt{KI}z)) \\
&= \lambda z.(\ulcorner \mathtt{S} \urcorner z(\ulcorner \mathtt{I} \urcorner z)\mathtt{I})(\ulcorner \mathtt{S} \urcorner z(\ulcorner \mathtt{I} \urcorner z)\mathtt{I}) \\
&= \lambda z.(\mathtt{KS}z(\mathtt{KI}z)\mathtt{I})(\mathtt{KS}z(\mathtt{KI}z)\mathtt{I}) \\
&= \lambda z.\mathtt{SII(SII)}
\end{aligned}$$

is unsolvable. By Proposition 8.2, $\ulcorner \cdot \urcorner$ does not satisfy Z. Then it also fails to satisfy DES, and is therefore not complete. ◀

Scott's coding is very elegant and minimalistic: the size of $\ulcorner M \urcorner$ is exactly double the size of M, and the number of strong (combinatory) reductions required to bring $\ulcorner M \urcorner\mathtt{I}$ to M is exactly the size of M. It is a shame that this coding is not complete. Could there be a substitute?

▶ **Definition 17.** Let the *minimal coding* be defined by

- $\ulcorner b \urcorner = \mathtt{SI}(\mathtt{K}b) = \underline{b}$, $b \in \{\mathtt{K}, \mathtt{S}\}$.
- $\ulcorner MN \urcorner = \mathtt{S}\ulcorner M \urcorner \ulcorner N \urcorner$

Note that $\ulcorner M \urcorner = \lambda z.M_z$, where M_z is obtained from M by replacing every occurrence of a basic combinator b with zb.

▶ **Proposition 18.** *The minimal coding is complete.*

Proof. The recursive definition makes it straightforward to construct $\ulcorner M \urcorner$ once the standard code of M is given:

$$U^{-1}(\underline{M}) = \text{IF } Z?\underline{M} \text{ THEN } \underline{M} \text{ ELSE } \mathtt{S}(U^{-1}(P_0\underline{M}))(U^{-1}(P_1\underline{M}))$$

(with P_i referring to the standard projections.)

For the opposite direction, we use the characterization result to infer the existence of a combinator satisfying U. By Corollary 14, having already defined U^{-1}, all that remains is to satisfy DES.

Let W and Z be the following terms:

$W = \lambda wnm.[w, [m\mathtt{S}, n\mathtt{S}]]$

$Z = \lambda x.[W, \mathtt{K}\langle x\rangle]$

By induction we will show that

$$\ulcorner M\urcorner Z = [W, M'], \qquad \text{with } M'\mathtt{S} = \ulcorner M\urcorner \tag{2}$$

Base case: If $b \in \{\mathtt{K}, \mathtt{S}\}$, then $\ulcorner b\urcorner Z = Zb = [W, \mathtt{K}\langle b\rangle]$. Furthermore,

$$\mathtt{K}\langle b\rangle\mathtt{S} = \langle b\rangle = \ulcorner b\urcorner$$

Induction: We compute

$$\begin{aligned}
\ulcorner MN\urcorner Z &= \mathtt{S}\ulcorner M\urcorner\ulcorner N\urcorner Z\\
&= \ulcorner M\urcorner Z(\ulcorner N\urcorner Z)\\
&=_{\mathrm{IH}} [W, M'][W, N']\\
&= [W, N']WM'\\
&= WWN'M'\\
&= [W, [M'\mathtt{S}, N'\mathtt{S}]]\\
&=_{\mathrm{IH}} [W, [\ulcorner M\urcorner, \ulcorner N\urcorner]]
\end{aligned}$$

Furthermore,

$$[\ulcorner M\urcorner, \ulcorner N\urcorner]\mathtt{S} = \mathtt{S}\ulcorner M\urcorner\ulcorner N\urcorner = \ulcorner MN\urcorner$$

Note that, with (2) verified, the proof of the induction step also gives us that $\ulcorner MN\urcorner Z = [W, [\ulcorner M\urcorner, \ulcorner N\urcorner]]$. Hence we can satisfy P by taking the terms $P_i = \lambda c.cZ\mathtt{U}^1_1\mathtt{U}^1_i$.

We can also separate application nodes from the leaves by the term

$$Z_?c = cZ\mathtt{U}^1_1(\lambda mno.\overline{\mathtt{K}})(\mathtt{KK})$$

Finally, $Z_?$ can be used to satisfy Z exactly as in the case of standard coding, by appealing to Böhm's theorem. (Using the fact that $\ulcorner\mathtt{K}\urcorner$ and $\ulcorner\mathtt{S}\urcorner$ are distinct, closed $\beta\eta$-normal forms.) ◀

▶ **Remark 19.** The reader might wonder whether the minimal coding could be trimmed even further by putting $\ulcorner b\urcorner = \mathtt{SI}b$ for the basic combinators. It turns out that this coding is complete as well, and for the uncoding map one can take $Ux = xZ\mathtt{U}^1_1$, where $Zx = [W, x(\mathtt{K}^4\underline{\mathtt{I}})(\mathtt{K}^3\underline{\mathtt{S}})(\mathtt{K}^2\underline{\mathtt{K}})]$ and $Wwnm = [w, [m, n]]$. However, since the evaluator is significantly more complex, we propose to regard the previous definition as *the* minimal quote for combinators.

References

1 Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics.* North-Holland, Amsterdam, 2nd edition, 1984.

2 Henk Barendregt. Discriminating Coded Lambda Terms. In A. Anderson and M. Zeleny, editors, *Logic, Meaning, and Computation: Essays in Memory of Alonzo Church*, volume 305 of *Synthese Library.* Springer, 1994.

3 Henk Barendregt. Enumerators of Lambda Terms Are Reducing Constructively. In *Dirk van Dalen Festschrift*, volume 5 of *Questionnes Infinitae*. Utrecht University, Department of Philosophy, 1999.

4 Henk Barendregt, June 2011. Private communication.

5 Alessandro Berarducci and Corrado Böhm. A self-interpreter of lambda calculus having a normal form. In Egon Börger, Gerhard Jäger, Hans Kleine Büning, Simone Martini, and Michael M. Richter, editors, *CSL*, volume 702 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 1992.

6 Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid.* Basic Books, 20 anv edition, February 1999.

7 Stephen C. Kleene. Lambda-definability and recursiveness. *Duke Mathematical Journal*, 2:340–353, 1936.

8 John McCarthy. *LISP 1.5 Programmer's Manual.* The MIT Press, 1962.

9 Torben Mogensen. Efficient Self-Interpretation in Lambda Calculus. *Journal of Functional Programming*, 2:345–364, 1994.

10 Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *PLDI*, pages 199–208, 1988.

11 Brian Cantwell Smith. *Procedural Reflection in Programming Languages, Volume I.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1982.

12 Raymond Smullyan. *Diagonalization and Self-Reference.* Oxford University Press, USA, 1994.