Juggling with combinators

Henk Barendregt
Computer Science Department
Catholic University
Nijmegen, The Netherlands

At this special day in honour of professor van der Poel it is appropriate to speak about one of his scientific interests: combinators. I will try to do this in the style of one of his more private interests, namely that of magic. By the way, nothing is mysterious about magic. By making appropriate use of the phenomena one can obtain unexpected results. This can be done with cards but, as we will see, also with combinators.

Nothing in this paper is new. Emphasis is on unexpected properties of combinators. If proofs are deleted, then they can be found in Barendregt [1984].

1. Rules of the game

The rules are simple. There are certain objects called *combinators*. These are built up from two basic combinators, **K** and **S** using a binary operation called *application*. If A and B are combinators, then so is AB, the result of *applying* A *to* B. Intuitively we can think of A as a function and of B as the argument. There are no other combinators than those built up from **K** and **S** using application.

Application will not be associative: in general $(AB)C \neq A(BC)$. We will use the convenient convention of association to the left: ABC denotes (AB)C and ABCD denotes ((AB)C)D; etcetera.

Now that we have the combinators and some notation for them, we can state the basic axioms that they satisfy.

1.1 Axioms of combinatory logic. For all combinators A, B and C one has

(1) $\mathbf{K}AB = A$;

(2) $\mathbf{S}ABC = AC(BC)$;

(3) $\mathbf{K} \neq \mathbf{S}$.

The theory axiomatised by these axioms is called combinatory logic, notation CL. It is known that this theory is consistent.

1.2 Theorem (Curry). Combinatory logic is consistent.

What is more, is that the theory is highly undecidable.

1.3 Theorem (Grzegorczyk). The equational theory combinatory logic is essentially

undecidable.

This means that given a consistent extension T of CL, then it is not decidable whether an equation A = B is derivable in T. In particular this also holds for T = CL.

One may wonder how such a simple axiom system 1.1 gives rise to undecidability. In section 3 we wiil see that combinators form a universal computation model. Therefore CL and consistent extensions are subject to the unsolvability of the halting problem.

2. Control

An important aspect of combinators is that they provide control of application. The following is an example.

2.1 Trick. There are combinators **I**, A and B such that

$$\mathbf{I}X = X;$$

$$AXY = Y;$$

$$BX = XX.$$

for all combinators X and Y.
Proof. Take $\mathbf{I} = \mathbf{SKK}$, $A = \mathbf{SK}$ and $B = \mathbf{S(SKK)(SKK)}$.
Then e.g. $\mathbf{I}X = \mathbf{SKK}X = \mathbf{K}X(\mathbf{K}X) = X;$
$AXY = \mathbf{SK}XY = \mathbf{K}Y(XY) = Y.$ □

The combinator **I** will be encounterd more often.

How does this trick work? In order to understand it we introduce so called *open combinators*, containing variables and define an abstraction operator on these.

2.2. Definition. (i) The set $\mathcal{X}$ of open combinators is defined as follows.

$$\mathbf{K}, \mathbf{S} \in \mathcal{X};$$

$$x_0, x_1, x_2, \ldots \in \mathcal{X};$$

$$A, B \in \mathcal{X} \Rightarrow (AB) \in \mathcal{X}.$$

(ii) If A is an open combinator, then the set of variables in A, notation FV(A), is defined as follows.

$$FV(\mathbf{K}) = FV(\mathbf{S}) = \emptyset;$$

$$FV(x_i) = \{x_i\};$$

$$FV(AB) = FV(A) \cup FV(B).$$

Ordinary combinators are open combinators A with $FV(A) = \emptyset$. These are also called *closed* combinators.

(iii) An open combinator equation A=B is called *valid* if after any substitution of closed combinators for the variables, we obtain an equation $A^S = B^S$ that is derivable from the axioms 1.1. For example $\mathbf{S}(\mathbf{K}xy) = \mathbf{S}x$ is valid. We may think of valid equations as being derived from the axioms 1.1, but with A, B now ranging over open combinators.

Instead of the variables $x_0, x_1, x_2, \ldots$ we often will write x, y, z, ... Outermost parentheses are omitted. So for example S(Kx)(Ky) is an open combinator and

$FV(S(Kx)(Ky)) = \{x,y\}$.

2.3 Theorem. For every open combinator P there exists an open combinator $\lambda x.P$ such that

1. $(\lambda x.P)x = P$ is valid;

2. $FV(\lambda x.P) = FV(P) - \{x\}$.

Proof. Induction on the structure of P. If P=x, then take $\lambda x.P = \mathbf{I}$. If P does not contain x, then take $\lambda x.P = \mathbf{K}P$. If P does contain x and is of the form QR, then take $\lambda x.P = S(\lambda x.Q)(\lambda x.R)$. □

2.4 Corollary. For every open combinator P and every sequence of variables $\vec{x} = x_1,...,x_n$ there exists an open combinator F such that

1. $F\vec{x} = P$ is valid;

2. $FV(F) = FV(P) - \{\vec{x}\}$.

In particular, if $FV(P) = \{\vec{x}\}$, then F is a (closed) combinator.
Proof. Take $F = \lambda x_1(\lambda x_2 ... (\lambda x_n.P)...)$. Then using 2.3 n times it follows that

$Fx_1x_2...x_n = P$ and $FV(F) = FV(P) - \{x_1,...,x_n\}$. □

Notation $\lambda\vec{x}.P = \lambda x_1...x_n.P = \lambda x_1.(\lambda x_2. ... (\lambda x_n.P)$. This is the explanation of the trick in 2.1. Indeed $A = \lambda xy.y$, $B = \lambda x.xx$ is another solution for 2.1.

3. Self reproduction

However, one can do much better.

3.1 Trick. There are combinators F and G such that for all combinators X and Y one has

$FXY = XG(YF)$;
$GX = XFG$. □

In this section we show how such combinators can be found.

3.2 Fixed-point theorem. For every combinator F there exists a combinator A such that

$FA = A$.

Proof. Using 2.4 find a combinator D such that

$Dx = F(xx)$.

Take $A = DD$. Then

$A = DD = F(DD) = FA$. □

3.3 Theorem. There exists a fixed-point combinator **Y** such that the fixed-points can be found uniformly:

$A = \mathbf{Y}F \Rightarrow FA = A$.

Proof. Take $\mathbf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$. □

If we write $P[\vec{x}]$, this means that P is an open combinator with free variables among the $\{x\}$. Then $P[\vec{A}]$ is the result of substituting the $\vec{A}$ for the $\vec{x}$ in P.

3.4 Corollary. Let $C[\vec{x}, f]$ be given. Then there exists a combinator F such that

$Fx = C[\vec{x}, F]$.

Proof. Take $F = \mathbf{Y}(\lambda f\vec{x}.C[\vec{x}, f])$. □

For example we can find an F such that
$$FX = XF.$$

3.5 Double fixed-point theorem. Given F, G there exists A, B such that
$$FAB = A$$
$$GAB = B.$$
Proof (Smullyan). By 3.4 let N satisfy
$$Nxyz = x\,(Nyyz)(Nzyz).$$
Take $A = NFFG$ and $B = NGFG$. □

3.6 Corollary. Given $C[\vec{x}, f, g]$ and $D[\vec{y}, f, g]$. Then one can find F and G such that
$$F\,x = C[\vec{x}, F, G]$$
$$G\,y = D[\vec{y}, F, G].$$
Proof. We'd like
$$F = \lambda\vec{x}.C[\vec{x}, F, G] = (\lambda fg\vec{x}.C[\vec{x}, f, g])FG,$$
$$G = \lambda\vec{y}.D[\vec{y}, F, G] = (\lambda fg\vec{y}.D[\vec{y}, f, g])FG.$$
Such F, G exist by 3.5. □

Using corollary 3.6 we can understand how to find the F and G in trick 3.1.
The double fixed-point theorem and its corollary can easily be generalized to n-fold versions.

4. Computing

In sections 2 and 3 we have seen that combinators give control over symbolic manipulations. A fortiori we can obtain control over arithmetic computations.
In order to do this, we have to introduce special combinators for truth values, the conditional, ordered pairs and numerals. Our representation of numerals is motivated by its simplicity. It is not efficient: in order to represent a number n, a combinator of complexity O(n) is needed. A more efficient way of doing numerical computations using combinators is given in van der Poel, Schaap and van de Mey [1980], where one of the representations of number n is of complexity O(log n).

4.1 Definition (truth values). Define
$$\mathbf{true} = K \text{ and } \mathbf{false} = KI.$$
Then $\mathbf{true}\,XY = X$ and $\mathbf{false}\,XY = Y$.

4.2 Definition (conditional).
The conditional
$$\mathbf{if}\ B\ \mathbf{then}\ X\ \mathbf{else}\ Y$$
can be represented by
$$BXY.$$
Indeed, if $B = \mathbf{true}$, then $BXY = X$ and if $B = \mathbf{false}$, then $BXY = Y$.

4.3 Definition (ordered pairs).
Ordered pairs of combinators can be represented as follows.

$[X,Y] = \lambda z.zXY.$

One can reconstruct both components from the pair

$[X,Y]$ **true** $= X,$

$[X,Y]$ **false** $= Y.$

4.4 Definition (numerals). Define

$\underline{0} = \mathbf{I}$

$\underline{n+1} = [\mathbf{false}, \underline{n}]$

4.5 Definition. Let f: N→N be a numerical function. Then f can be *represented by the conbinator* F if for all n

$F\,\underline{n} = \underline{f(n)}.$

Similarly representability for functions of more arguments is defined; e.g. a function g of two arguments is represented by a combinator G if

$G\,\underline{n}\,\underline{m} = \underline{g(n,m)}.$

4.6 Lemma. There are combinators S_+, P_-, $Z_?$ for successor, predecessor and test for zero such that for all n

$S_+\,\underline{n} = \underline{n+1},$

$P_-\,\underline{n+1} = \underline{n},$

$Z_?\,\underline{0} = \mathbf{true},$

$Z_?\,\underline{n+1} = \mathbf{false}.$

Proof. Take $S_+ = \lambda x.[\mathbf{false},x],$

$P_- = \lambda x.x\mathbf{false},$

$Z_? = \lambda x.x\mathbf{true}.$

4.7 Theorem (Kleene). Let f be a computable function. Then f can be represented by a combinator.

Proofsketch. We show how to deal with recursion and minimalization. These are the essential algorithmic components for obtaining the class of computable functions.

Recursion. Let f be defined by

$f(0) = 13$

$f(n+1) = g(f(n)),$

where g is a known function that can be represented by the combinator G. This is a simple version of recursion, called iteration, but the method of representation in typical. One can rewrite f as follows.

$f(x) =$ **if** $x=0$ **then** 13 **else** $g(f(x-1)).$

Now the representation of this f is simply a combinator F such that

$Fx =$ **if** $Z_?x$ **then** $\underline{13}$ **else** $G(F(P_x)).$

Such an F exists as we saw in section 3.

Minimalisation. Let b be a given computable binary predicate on numbers. We want to compute

$f(x) = \mu y.\ b(x,y)$

i.e. the least y such that b(x,y) holds.

We may assume that b is already represented by the combinator B, that is

$b(n,m) \Leftrightarrow B\,\underline{n}\,\underline{m} = \mathbf{true}$

Now f can be represented by F satisfying

$Fx = Hx\underline{0}$

where hxy = **if** Bxy **then** y **else** $Bx(S_+y)$. □

4.8 Trick. Let $x^{\sim n}$ be a sequence of n times x. There exists a combinator P such that

$P(\lambda x.x^{\sim n})$ = **true** if n is a prime number;
= **false** else.

The solution can be found in van der Poel et al [1980], where the $\lambda x.x^{\sim n}$ are used as numerals.

5. Algebras

The solution of trick 4.8 makes use of the combinator
M = **SI**
that satisfies the following partial associative laws:

MMM = M(MM),

MMMM = M(M(MM)),

etcetera.

Note however that (MM)(MM) ≠ MMMM. The LHS has no normal form, but the RHS does. So the associativity is not complete.

The combinator is called M because G. van der Mey proved that

MY = Y ⇒ YF = F(YF).

That is, every fixedpoint of M is a fixedpoint operator.

One may wonder whether there are fully associative combinators.

5.1 Trick. (i) There exist distinct combinators M_e, M_a, M_b, M_c such that these behave like the Klein fourgroup {e, a, b, c}:

$M_eM_a = M_aM_e = M_a$, $M_aM_a = M_e$ etcetera.

(ii) One can also embed a monster group.

The solution shows that the examples are somewhat exagerated.

5.2 Proposition (Barendregt, Dezani and Klop).
Given is a computable applicative structure (A, .). That is, A is a countable set, say ℕ the set of natural numbers, and . is a computable binary operation on A. Then (A, .) can be embedded isomorphically into the combinators.
Proof. Let A=ℕ and let a.b = f(a,b) be a computable function. Let the combinator F represent f. Define.

$M_a = [G,\underline{a}]$,

where G is still to be determined. Compute

$M_aM_b = [G.\underline{a}][G,\underline{b}]$
$= [G,\underline{b}]\ G\ \underline{a}$
$= GG\underline{b}\ \underline{a}$

If we take G = λpqr.[p,Frq], then this computation continues as follows.

$= [G, F\ \underline{a}\ \underline{b}]$
$= [G, \underline{f(a,b)}] = M_{f(a,b)}$.

Moreover the M_a's are all distinct. Therefore $H(a) = M_a$ is the required embedding. □

By some cardinality argument applied to representable functions it can be shown that not every countable applicative structure can be embedded in the combinators.
It is however possible to embed arbitrary applicative structures in so called combinatory algebras.

5.3 Definition. A *combinatory algebra* is an applicative structure (A, ., K, S) with K, S∈ A such that the axioms 1, 2 and 3 of 1.1. hold.

The following is proved in Engeler [1981].

5.4 Proposition. Given an applicative structure (A, .). Then there exists a combinatory algebra D_A such that (A, .) can be embedded isomorphically into D_A.

6. Equations

An equation P=Q between two combinators is called *consistent* if from P=Q one cannot derive **K=S**. Equivalently, if P=Q is true in some combinatory algebra.
For example
K=KKK is consistent,
I=S is inconsistent.
The first equation is derivable. From **I=S** one derives **IKKS = SKKS**, hence **K=S.**

Tnere are also 'independent' equations, i.e. consistent ones that cannot be derived.

6.1 Proposition. Let **Ω = SII(SII)**. Then for every combinator P one has that Ω=P is consistent.
Proof. Jacopini [1975] has shown this by a proof theoretic argument. Baeten end Boerboom [1979] have shown this by constructing combinatory algebras. □

6.2 Trick (Bel). There exists a magic triple, that three combinators P, Q, R such that
P=Q is consistent
Q=R is consistent
R=P is consistent
but P=Q=R is not consistent.

Proof. Take P=**I**, Q=Ω and R=Ω**S**.
That **I**=Ω is consistent follows from 6.1. Similarly Ω=Ω**S** is consistent, since the equation **I** = Ω**S** follows from Ω=**KI** and we can use 6.1.
But **I**=Ω=Ω**S** implies **I**=**IS**=**S** and we have seen that this equation is inconsistent. □

M. Bel has generalised his trick to magic n-tuples.

References

Barendregt, H.P.
[1984] *The lambda calculus, its syntax and semantics*, North Holland, Amsterdam.

Baeten, J. and B.Boerboom
[1979] Ω can be anything it shouldn't be, *Indag.Math.* 41, 111-120.

Engeler, E.
[1981] Algebras and combinatory, *Algebra universalis* 13 no. 3, 289-392.

Jacopini, G.
[1975] A condition for identifying two elements of whatever model of combinatory logic, Springer LMCS 37, 213-219.

van der Poel, W.L., C.E. Schaap and G. van der Mey
[1980] New arithmetical operators in the theory of combinators, *Proc. Kon. Ned. Ak. Wetensch.*, Series A, vol. 83 no. 3, 271-325.