CS263–Spring 2008
Topic 1: The Lambda Calculus
Section 2.2: Undecidability

Dana S. Scott
Hillman University Professor (Emeritus)
School of Computer Science
Carnegie Mellon University
====
Visiting Professor EECS
Visiting Scientist
Logic & Methodology Program
University of California, Berkeley

Last edited 1 February 2008

Undecidability

Setting things up

Combinator definitions

Here is the total suite of ***definitions*** we have been using (plus a couple of extra).

Marked in ***yellow*** are the *Mathematica* definitions for ***manipulating*** combinations.

The others are ***assignments*** for introducing short-hand names for ***specific combinators***.

```
crules = {J[x_] → x, S[x_][y_][z_] → x[z][y[z]], K[x_][y_] → x};
```

```
ToC[vars_, comb_] := Fold[rm, comb, Reverse[vars]];

rm[v_, v_] := J;
rm[f_[v_], v_] /; FreeQ[f, v] := f;
rm[h_, v_] /; FreeQ[h, v] := K[h];
rm[f_[g_], v_] := S[rm[f, v]][rm[g, v]];
```

```
Y = ToC[{f}, ToC[{x}, f[x[x]]][ToC[{x}, f[x[x]]]]];
```

```
zero = K[J];
succ = S[S[K[S]][K]];
plus = S[K[S]][S[K[S[K[S]]]][S[K[K]]]];
times = S[K[S]][K];
power = S[K[S[J]]][K];
```

```
one = succ[zero];
two = succ[one];
three = succ[two];
four = succ[three];
```

```
sum = ToC[{n, m}, n[succ][m]];
prod = ToC[{n, m}, n[sum[m]][zero]];
exp = ToC[{n, m}, m[prod[n]][one]];
```

```
cnum[0] := zero;
cnum[n_] := succ[cnum[n - 1]];
```

```
pair = ToC[{x, y, z}, z[x][y]];
```

```
left = ToC[{x, y}, x];
right = ToC[{x, y}, y];
```

```
shift = ToC[{p}, pair[succ[p[left]]][p[left]]] //. crules;
pred = ToC[{n}, n[shift][pair[zero][zero]][right]] //. crules;
```

```
shift1 = ToC[{p}, pair[p[right]][right]] //. crules;
zeroQ = ToC[{n}, n[shift1][pair[left][right]][left]] //. crules;
```

```
equalQ = ToC[{n, m}, pair[zeroQ[m[pred][n]]][right][zeroQ[n[pred][m]]]];
```

■ Gödel numbers of combinations

Stop to think for a moment about the fact that every ***positive*** integer **n** can be ***uniquely factored*** as $2^p\,(2\,q+1)$.

We can use this fact to assign to every ***pure combinator*** (*i.e.*, without variables) a unique ***Gödel number***.

```
godel[J] := 0;
godel[S] := 1;
godel[K] := 2;
godel[a_[b_]] := 2 + 2^godel[a] (2 godel[b] + 1);
```

In the ***other direction***, we can ask of a number ***which combinator*** has it as a Gödel number?

```
combo[0] := J;
combo[1] := S;
combo[2] := K;
combo[n_] := Block[{p = IntegerExponent[n - 2, 2]},
   Block[{q = 1/2 ((n - 2)/2^p - 1)}, combo[p][combo[q]]]];
```

Note that **godel** and **combo** are *Mathematica* programs and ***not*** combinators.

Here are some tests of these *Mathematica* programs.

```
combo[987 654 321]
```

```
J[J[J[J][J[S][K[S[K[J[J[J[J[S[J[J[S]]]]]]]]]]]]]]
```

```
godel[combo[987 654 321]]
```

```
987 654 321
```

```
godel[S[S[S[K]]]]
```

```
212
```

```
combo[212]
```

```
S[S[S[K]]]
```

```
combo[555]
```

```
J[S[S[S[J[J]]]]]
```

```
combo[2 + 2^212 (2 × 555 + 1)]
```

```
S[S[S[K]]][J[S[S[S[J[J]]]]]]
```

```
combo[2 + 2^555 (2 × 212 + 1)]
```

```
J[S[S[S[J[J]]]]][S[S[S[K]]]]
```

An undecidability proof

Theorem. There is no effective ***decision procedure*** for deciding whether a given combinator reduces to a ***normal form*** or not.

Note that these two combinators are ***different*** and ***do not reduce*** to each other.

```
left
right
```

```
K
```

```
K[J]
```

If it were ***decidable*** by recursive functions whether ***in every case*** a combinator has a normal form, then we could find a combinator **halt** such that for all integers **n** and **m**:

```
halt[cnum[n]][cnum[m]] ⇒ left
 ⟺ combo[n][cnum[m]] has a normal form
halt[cnum[n]][cnum[m]] ⇒ right
 ⟺ combo[n][cnum[m]] has no normal form
```

Let **N** be a (known) combinator ***without*** a normal form (nick-name it ***Nasty***), and take a look at this next combinator defined in terms of this hypothetical combinator **halt**:

```
ToC[{x}, pair[N][K][halt[x][x]]]
S[K[S[S[K[S]][S[K[K]][S[K[S]][S[K[S[J]]][K]]]]][K[K]][N][K]]][S[halt][J]]
```

Next, ***think*** of its Gödel number.

```
g == godel[ToC[{x}, pair[N][K][halt[x][x]]]]
```

Suppose first that:

```
combo[g][cnum[g]] has a nf
```

and so

```
halt[cnum[g]][cnum[g]] ⇒ left
```

This means that

```
ToC[{x}, pair[N][K][halt[x][x]]][cnum[g]] has a nf
```

In other words,

```
pair[N][K][halt[cnum[g]][cnum[g]]] has a nf
```

But this ***reduces at once*** to **N**, which has ***no*** normal form.

The ***contradiction*** proves:

```
combo[g][cnum[g]] has no nf
```

and so

```
halt[cnum[g]][cnum[g]] ⇒ right
```

But, as before

```
combo[g][cnum[g]] ⇒ pair[N][K][halt[cnum[g]][cnum[g]]]
                  ⇒ pair[N][K][right] ⇒ K
```

That is, **combo[g][cnum[g]]** ***has*** a normal form, which is ***imposible***.

Thus, there is ***no such*** combinator as **halt**, and there can be ***no decision method*** for normal forms. **Q.E.D.**