CS263-Spring 2008
Topic 1: The Lambda Calculus
Section 3.2: Semantics II

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 9 February 2008

Computable Operators

Defining computability

- **Those combinators (in case we need them)**
- **Sequences and operators (in case we need them)**
- **Using recursive enumerability**

The ***recursively enumerable*** sets represent what is computable among sets of integers — at least ***in theory***. We now use them to ***define*** what we mean by a ***computable operator***.

A continuous operator $\Phi(X_0, X_1, \ldots, X_{n-1})$ is said to be ***computable*** if, and only if,

$$\lambda X_0.\lambda X_1.\ldots.\lambda X_{n-1}.\Phi(X_0, X_1, \ldots, X_{n-1}) \in \mathbb{RE}$$

Some examples are needed, which have to be worked out from the definition of the ***λ-abstraction*** process.

Theorem. Writing $J = \lambda X.X$ and $K = \lambda X.\lambda Y.X$ and $K[J] = \lambda Y.\lambda X.X$, we find:

$J = \{0\} \cup \{(x, n) \mid n \in \S x\}$,

$K = \{0\} \cup \{(x, 0) \mid x \in \mathbb{N}\} \cup \{((x, (y, n)) \mid n \in \S x\}$, and

$K[J] = \{0\} \cup \{(y, 0) \mid y \in \mathbb{N}\} \cup \{((y, (x, n)) \mid n \in \S x\}$.

Note. The above sets are all ***recursive***.

Question. What about $S = \lambda X.\lambda Y.\lambda Z.X[Z][Y[Z]]$? Exercise?

Here is another important one, which brings in ***application***:

Theorem. Writing $Ap = \lambda U.\lambda X.U[X]$, we find:

$$Ap = \{0\} \cup \{(u, 0) \mid u \in \mathbb{N}\} \cup \{((u, (x, n)) \mid \exists (x_1, n) \in \S u.\S x_1 \subseteq \S x\}.$$

Note. It is another ***recursive*** set. (Why?)

The stuff seen above with all those 0s is not so important. The definition comes down to something simpler:

Theorem. A continuous operator $\Phi(X_0, X_1, \ldots, X_{n-1})$ is ***computable***

if, and only if,

$$\{(x_0, (x_1, \ldots, (x_{n-1}, m) \ldots)) \mid m \in \Phi(\S x_0, \S x_1, \ldots, \S x_{n-1})\} \in \mathbb{RE}$$

This means that you have a recursively enumerable way of generating how ***output from the operator depends on (finite) inputs***.

Note. In the recursive-function literature, the computable, continuous operators are called "***enumeration operators***". Unfortunately that literature does not relate so directly to Computer Science.

■ ℝ𝔼 as a model for combinators

From the definition of $U[X]$ we can see that in case $U \in \mathbb{RE}$, then if $X \in \mathbb{RE}$, so is $U[X]$. This observation is important enough to call a theorem, which has several corollaries:

Theorem. Not only is $U[X]$ a ***computable operator*** (of two arguments), but the family $\mathbb{RE}$ is closed under ***application***.

Corollary. Computable operators (of any number of arguments) are closed under ***composition***. (Why?)

Now, since the sets corresponding to **J**, **K**, and **S** are indeed in $\mathbb{RE}$, we see at once we have another model for the combinators.

Corollary. $\mathbb{RE}$ is a ***model*** for the **cRules** of combinators using the operator $U[X]$ as the ***application operation***.

Note. The models $\mathbb{P}$ and $\mathbb{RE}$ both have the property that the **cRules** are satisfied as ***equations***, not just ***reduction rules***:

$J[X] = X$
$K[X][Y] = X$
$S[X][Y][Z] = X[Z][Y[Z]]$

Question. Are there ***other*** models for the **`crules`** between $\mathbb{RE}$ and $\mathbb{P}$ also using the same $\boldsymbol{U[X]}$? Exercise?