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,
Some examples are needed, which have to be worked out from the definition of the \textit{\lambda\textit{-abstraction}} process.

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

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

\textbf{Note.} The above sets are all \textit{recursive}.

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

Here is another important one, which brings in \textit{application}:

\textbf{Theorem.} Writing $Ap = \lambda U.\lambda X.\lambda Y.\lambda Z.X[Z][Y[Z]]$, we find:

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

\textbf{Note.} It is another \textit{recursive} set. (Why?)

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

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

if, and only if,

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

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

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

\section{\textit{\mathbb{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:

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

\textbf{Corollary.} Computable operators (of any number of arguments) are closed under \textit{composition}. (Why?)

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

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

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

- $J[X] = X$
- $K[X][Y] = X$
- $S[X][Y][Z] = X[Z][Y[Z]]$
Question. Are there other models for the rules between $\mathbb{R}[E]$ and $P$ also using the same $U[X]$? Exercise?