8 Numbering computable functions

In chapter 2.2 we studied codings of words into numbers, which allow us to code any (text of a) program into a number. In this chapter we study some consequences of this coding. The most important results are:

**Kleene's Normal Form Theorem:** Every partial recursive function can be defined in a very simple form involving only one application of minimization.

**Turing's Theorem on Universal Machines:** There exists a Turing machine able to simulate all Turing machines.

**Kleene's Recursion Theorem:** The partial recursive definition are closed under arbitrary recursive definitions.

As we will see, Kleene's recursion theorem provides additional evidence for the validity of the Church-Turing Thesis.

8.1 Numbering Turing machines

In the following we consider Turing machines of the form

\[(\{0, 1, \square\}, Q, P)\]

only, where \(Q\) contains a distinguished initial state \(q_0\). Remember that these kind of Turing machines were used in section 4.2 to define Turing computable functions.

We define a trivial Turing machine by

\[M_{\text{triv}} := (\{0, 1, \square\}, \{q_0\}, \emptyset)\]

Since \(M_{\text{triv}}\) does not perform a single computation step we have

\[M_{\text{triv}}^{(k)}(x_1, \ldots, x_k) = x_1\]

For any number \(e \in \mathbb{N}\) we set

\[M_e := \begin{cases} 
\text{the Turing machine coded by } e & \text{if } e \text{ codes a Turing machine} \\
\text{the trivial Turing machine } M_{\text{triv}} & \text{otherwise}
\end{cases}\]

If \(M = M_e\) we call \(e\) a **Gödel number** of \(M\). By construction the sequence

\[M_0, M_1, M_2, \ldots\]

is a listing of all Turing machines.

We set

\[\{e\}^{(k)} := M_e^{(k)}\]

i.e., \(\{e\}^{(k)}\) is the partial recursive function computed by \(M_e\), also called the **partial recursive function with Gödel number** \(e\). Hence for \(e, x_1, \ldots, x_k \in \mathbb{N}\) we have

\[\{e\}^{(k)}(x_1, \ldots, x_k) \simeq M_e^{(k)}(x_1, \ldots, x_k)\]

For every partial recursive function \(f : \mathbb{N}^k \rightarrow \mathbb{N}\) there is a Turing machine \(M_e\) computing it, that is, \(\{e\}^{(k)} = f\). The number \(e\) is called a **Gödel number of** \(f\). Obviously for every partial recursive function \(f\) there are infinitely many Turing machines computing \(f\) (just modify a Turing machine for \(f\) by adding dummy instructions). Consequently \(f\) has infinitely many Gödel numbers.
8.2 The S-m-n Theorem

If \( f \) is a partial recursive function of \( m + n \) arguments, and we fix \( m \) arguments of \( f \), then we obtain a function of \( n \) arguments which again is partial recursive. The following theorem says that we can compute a Gödel number of the new function from a Gödel number of \( f \) and the fixed arguments in a primitive recursive way.

**S-m-n Theorem.**

For all \( m, n \in \mathbb{N} \) there is a primitive recursive function \( S^m_n: \mathbb{N}^{m+1} \to \mathbb{N} \) such that

\[
\{S^m_n(e, x_1, \ldots, x_m)\}^{[n]}(y_1, \ldots, y_n) \simeq \{e\}^{[m+n]}(x_1, \ldots, x_m, y_1, \ldots, y_n)
\]

for all \( e, x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{N} \)

**Sketch of proof.** Given a Turing machine \( M \) and numbers \( x_1, \ldots, x_m \) it is easy to construct a new Turing machine \( M' \) such that for all \( y_1, \ldots, y_n \in \mathbb{N} \) we have

\[
M'^{[n]}(y_1, \ldots, y_n) \simeq M^{[m+n]}(x_1, \ldots, x_m, y_1, \ldots, y_n)
\]

The function \( S^m_n \) now performs the construction of \( M' \) from \( M \) and \( x_1, \ldots, x_m \) on the level of codes, that is, if \( M = M_e \) then

\[
M' = M_{e'}, \text{ where } e' = S^m_n(e, x_1, \ldots, x_m)
\]

It is fairly easy to see that the function \( S^m_n \) can be defined primitive recursively.

8.3 Kleene’s Normal Form Theorem

The following theorem, due to S.C. Kleene, states that the partial recursive functions can be computed in a uniform way from two primitive recursive functions and on application of minimization.

**Notation.** Let \( R \) be a predicate of \( k + 1 \) arguments. We set

\[
\mu y. R(x_1, \ldots, x_n, y) := \left\{ \begin{array}{ll}
\text{the least } y \text{ s.t. } R(x_1, \ldots, x_n, y) & \text{if such } y \text{ exists} \\
\text{undefined} & \text{otherwise}
\end{array} \right.
\]

**Kleene’s Normal Form Theorem** For every \( k \) there are a primitive recursive relation \( T_k \) of \( k + 2 \) arguments and a primitive recursive function \( U \) of one argument, such that

\[
\{e\}^{[k]}(x_1, \ldots, x_k) \simeq U(\mu y. T_k(e, x_1, \ldots, x_k, y))
\]

**Sketch of proof.** We define \( T_k \) by

\[
T_k(e, x_1, \ldots, x_k, y) \text{ if and only if } y \text{ codes a pair } (t, c) \text{ such that } c \text{ is the configuration of } M^{(k)}_e(x_1, \ldots, x_k) \text{ after } t \text{ computation steps, and at that configuration } M^{(k)}_e(x_1, \ldots, x_k) \text{ halts.}
\]

\( U \) just reads off the result from a configuration.
8.4 Universal Turing machines

For every number \( k \in \mathbb{N} \) the **universal function** for the partial recursive functions of \( k \) arguments is the partial function \( \text{univ}_k : \mathbb{N}^{k+1} \to \mathbb{N} \) defined by

\[
\text{univ}_k(e, x_1, \ldots, x_k) \doteq \{ e \}^{(k)}(x_1, \ldots, x_k)
\]

Intuitively, \( \text{univ}_k \) is computable. Any Turing machine \( M \) computing \( \text{univ}_k \) is called a **universal Turing machine**. Hence, using the Church-Turing thesis, we know that universal Turing machines exist. However, using Kleene’s Normal Form Theorem and the equivalence of Turing-computability and partial recursiveness proven in section 7.1, we can prove the existence of universal Turing machines directly.

**Turing’s Theorem on Universal Machines**

For every \( k \) there exists a universal Turing machine for the partial recursive functions of \( k \) arguments.

**Proof.** Because of the equivalence theorem it suffices to show that \( \text{univ}_k \) is partial recursive. According to Kleene’s normal form theorem we have

\[
\text{univ}_k(e) \simeq U(\mu y. T_k(e, x_1, \ldots, x_n, y))
\]

which shows that \( \text{univ}_k \) is indeed partial recursive.

8.5 The Recursion Theorem

We now use the previous results to prove that partial recursive functions are closed under arbitrary recursions.

**Kleene’s Recursion Theorem**

Let \( f : \mathbb{N}^{n+1} \to \mathbb{N} \) be a partial recursive function. Then there exists \( e \in \mathbb{N} \) such that

\[
f(e, x_1, \ldots, x_n) \doteq \{ e \}^{(n)}(x_1, \ldots, x_n)
\]

for all \( x_1, \ldots, x_n \in \mathbb{N} \).

**Proof.** Define \( h : \mathbb{N}^{n+1} \to \mathbb{N} \) by

\[
h(y, x_1, \ldots, x_n) \doteq f(S^n_1(y, y), x_1, \ldots, x_n)
\]

Clearly \( h \) is partial recursive. Let \( d \) be a Gödel number of \( h \), that is,

\[
h = \{ d \}^{(n+1)}
\]

Set

\[
e := S^n_1(d, d)
\]
Now
\[
\begin{align*}
f(e, x_1, \ldots, x_n) & \overset{(3)}{=} f(S^n(d, d), x_1, \ldots, x_n) \\
& \overset{(1)}{=} h(d, x_1, \ldots, x_n) \\
& \overset{(2)}{=} (d)^{(n+1)}(d, x_1, \ldots, x_n) \\
& \overset{(S-m-n)}{=} (S^n(d, d))^{(n)}(x_1, \ldots, x_n) \\
& \overset{(3)}{=} [e]^{(n)}(x_1, \ldots, x_n)
\end{align*}
\]

**Application.** Recall the Ackermann function:

\[
\begin{align*}
\text{Ack}(0, y) &= y + 1 \\
\text{Ack}(x + 1, 0) &= \text{Ack}(x, 1) \\
\text{Ack}(x + 1, y + 1) &= \text{Ack}(x, \text{Ack}(x + 1, y))
\end{align*}
\]

We show that Ack is recursive. Since we know that Ack is total (\(\text{Ack}(x, y) = \text{Ack}_z(y)\)) it suffices to show that Ack is partial recursive.

Define \(f : \mathbb{N}^3 \rightarrow \mathbb{N}\) by case analysis:

\[
\begin{align*}
f(z, 0, y) & \equiv y + 1 \\
f(z, x + 1, 0) & \equiv \text{univ}_2(z, x, 1) \\
f(z, x + 1, y + 1) & \equiv \text{univ}_2(z, x, \text{univ}_2(z, x + 1, y))
\end{align*}
\]

Since \(\text{univ}_2\) is partial recursive, so is \(f\). By definition of the universal function, \(\text{univ}_2\), we have

\[
\begin{align*}
f(z, 0, y) & \equiv y + 1 \\
f(z, x + 1, 0) & \equiv [z]^{(2)}(x, 1) \\
f(z, x + 1, y + 1) & \equiv [z]^{(2)}(x, [z]^{(2)}(x + 1, y))
\end{align*}
\]

Now, by Kleene’s recursion theorem, there is a number \(e \in \mathbb{N}\) such that

\[
f(e, x, y) = [e]^{(2)}(x, y)
\]

for all \(x, y \in \mathbb{N}\). Therefore, setting \(z := e\) in the defining equations of \(f\) and applying the equation above we obtain

\[
\begin{align*}
[e]^{(2)}(0, y) & \equiv y + 1 \\
[e]^{(2)}(x + 1, 0) & \equiv [e]^{(2)}(x, 1) \\
[e]^{(2)}(x + 1, y + 1) & \equiv [e]^{(2)}(x, [e]^{(2)}(x + 1, y))
\end{align*}
\]

which shows that \([e]^{(2)}\) satisfies exactly the defining equations of the Ackermann function, Ack. Hence
\[ \text{Ack} = \{e\}^{(2)} \]

More formally one proves by induction on \( x \)

\[ \text{Ack}_x = \text{the function } y \mapsto \{e\}^{(2)}(x, y) \]

Therefore \( \text{Ack}(x, y) = \text{Ack}_x(y) = \{e\}^{(2)}(x, y) \) for all \( x, y \in \mathbb{N} \).

Since \( \{e\}^{(2)} \) is partial recursive, so is \( \text{Ack} \).