Chapter 2

Prefix-free codes

2.1 Domain of definition

Definition 2.1. *The* domain of definition *of the Turing machine T is the set*

$$\Omega(T) = \{p \in \mathbb{S} : T(p) \neq \perp\}.$$

In other words, $\Omega(T)$ is the set of strings p for which $T(p)$ is defined.

Recall that $T(p)$ may be undefined in three different ways:

Incompletion The computation may never end;

Under-run The machine may halt before reading the entire input string p;

Over-run There may be an attempt to read beyond the end of p.

We do not distinguish between these 3 modes of failure, writing

$$T(p) = \perp.$$

in all three cases.

As we shall see, this means that $T(p)$ can only be defined for a restricted range of input strings p. At first sight, this seems a serious disadvantage; one might suspect that Chaitin's modification of the Turing machine had affected its functionality. However, as we shall see, we can avoid the problem entirely by encoding the input; and this even turns out to have incidental advantages, for example extending the theory to objects other than strings.

2.2 Prefix-free sets

Definition 2.2. *Suppose $s, s' \in \mathbb{S}$. We say that s is a* prefix *of s', and we write $s \prec s'$, if s is an initial segment of s', ie if*

$$s' = b_0 b_1 \ldots b_n$$

then

$$s = b_0 b_1 \ldots b_r$$

for some $r \leq n$.

Evidently

$$s \prec s' \Longrightarrow |s| \leq |s'| .$$

Definition 2.3. *A subset $S \subset \mathbb{S}$ is said to be* prefix-free *if*

$$s \prec s' \Longrightarrow s = s'$$

for all $s, s' \in S$.

In other words, S is prefix-free if no string in S is a prefix of another string in S.

Theorem 2.1. *The domain of definition of a Turing machine T,*

$$\Omega(T) = \{s \in \mathbb{S} : T(s) \neq \perp\},$$

is prefix-free.

Proof ►. Suppose $s' \prec s,\ s' \neq s$. Then if $T(s')$ is defined, the machine must halt after reading in s, and so it cannot read in the whole of s. Hence $T(s)$ is undefined.

Proposition 2.1. *If $S \subset \mathbb{S}$ is prefix-free then so is every subset $T \subset S$.*

Proposition 2.2. *A prefix-free subset $S \subset \mathbb{S}$ is* maximal *(among prefix-free subsets of $\mathbb{S}$) if and only if each $t \in \mathbb{S}$ is either a prefix of some $s \in S$ or else some $s \in S$ is a prefix of t.*

Remark. For those of a logical turn of mind, we may observe that being prefix-free is a *property of finite character*, that is, a set S is prefix-free if and only if that is true of every finite subset $F \subset S$. It follows by Zorn's Lemma that each prefix-free set S is contained in a maximal prefix-free set. However, we shall make no use of this fact.

2.3 Prefix-free codes

Definition 2.4. *A* coding *of a set X is an injective map*

$$\gamma : X \to \mathbb{S}.$$

The coding γ is said to be prefix-free *if its image*

$$\operatorname{im} \gamma \subset \mathbb{S}$$

is prefix-free.

By encoding X, we can in effect take the elements $x \in X$ as input for the computation $T(\gamma x)$; and by choosing a prefix-free encoding we allow the possibility that the computation may complete for all $x \in X$.

2.4 Standard encodings

It is convenient to adopt *standard* prefix-free encodings for some of the sets we encounter most often, for example the set $\mathbb{N}$ of natural numbers, or the set of Turing machines. In general, whenever we use the notation $\langle x \rangle$ without further explanation it refers to the standard encoding for the set in question.

2.4.1 Strings

Definition 2.5. *We encode the string*

$$s = b_1 b_2 \cdots b_n \in \mathbb{S}.$$

as

$$\langle s \rangle = 1b_1 1b_2 1 \cdots 1b_n 0.$$

Thus a 1 in odd position signals that there is a string-bit to follow, while a 0 in odd position signals the end of the string.

Example. If $s = 01011$ then

$$\langle s \rangle = 10111011110.$$

If $s = \square$ (the empty string) then

$$\langle s \rangle = 0.$$

Definition 2.6. *We denote the length of the string $s \in \mathbb{S}$, ie the number of bits in s, by $|s|$.*

Evidently

$$|s| = n \implies |\langle s \rangle| = 2n + 1.$$

Proposition 2.3. *The map*

$$s \mapsto \langle s \rangle : \mathbb{S} \to \mathbb{S}$$

defines a maximal prefix-free code for $\mathbb{S}$.

Proof ▶. A string is of the form $\langle s \rangle$ if and only if

1. it is of odd length,
2. the last bit is 0, and
3. this is the only 0 in an odd position.

The fact that $\langle s \rangle$ contains just one 0 in odd position, and that at the end, shows that the encoding is prefix-free.

To see that it is *maximal*, suppose $x \in \mathbb{S}$ is not of the form $\langle s \rangle$ for any $s \in \mathbb{S}$. We need only look at the odd bits of x. If there is no 0 in odd position then appending 0 or 00 to x (according as x is of even or odd length) will give a string of form $\langle s \rangle$. If there is a 0 in odd position, consider the first such. If it occurs at the end of x then x is of form $\langle s \rangle$, while if it does not occur at the end of x then the prefix up to this 0 is of the form $\langle s \rangle$ for some s.

It follows that if x is not already of the form $\langle s \rangle$ then it cannot be appended to the set $\{\langle s \rangle : s \in \mathbb{S}\}$ without destroying the prefix-free property of this set.

2.4.2 Natural numbers

Definition 2.7. *Suppose $n \in \mathbb{N}$. Then we define $\langle n \rangle$ to be the string*

$$\langle n \rangle = \overbrace{1 \cdots 1}^{n \text{ 1's}} 0.$$

Example.

$$\langle 3 \rangle = 1110$$
$$\langle 0 \rangle = 0.$$

Proposition 2.4. *The map*

$$n \mapsto \langle n \rangle : \mathbb{N} \to \mathbb{S}$$

defines a maximal prefix-free code for $\mathbb{N}$.

2.4.3 Turing machines

Recall that a Turing machine T is defined by a set of rules

$$R : (q, b) \mapsto (a, q').$$

We encode this rule in the string

$$\langle R \rangle = \langle q \rangle b \langle a \rangle \langle q' \rangle,$$

where the 6 actions are coded by 3 bits as follows:

$$\begin{array}{ll} \texttt{noop} & \mapsto 000 \\ \texttt{swap} & \mapsto 001 \\ \longleftarrow & \mapsto 010 \\ \longrightarrow & \mapsto 011 \\ \texttt{read} & \mapsto 100 \\ \texttt{write} & \mapsto 101 \end{array}$$

So for example, the rule $(1, 1) \mapsto (\longleftarrow, 2)$ is coded as

$$1011010110.$$

Definition 2.8. *Suppose the Turing machine T is specified by the rules $R_1, \ldots, R_n$. Then we set*

$$\langle T \rangle = \langle n \rangle \langle R_1 \rangle \cdots \langle R_n \rangle.$$

We do not insist that all the rules are given, adopting the convention that if no rule is given for (q, b) then the 'default rule'

$$(q, b) \mapsto (\texttt{noop}, 0)$$

applies.

Also, we do not specify the order of the rules; so different codes may define the same machine.

2.4.4 Product sets

Proposition 2.5. *If S and S' are both prefix-free subsets of $\mathbb{S}$ then so is*

$$SS' = \{ss' : s \in S, s' \in S'\},$$

where ss' denotes the concatenation of s and s'.

Proof ▶. If $s_1 s'_1 \prec s_2 s'_2$ then either (a) $s_1 \prec s'_1$, or (b) $s'_1 \prec s_1$, or (c) $s_1 = s'_1$ and either $s_2 \prec s'_2$ or $s'_2 \prec s_2$.

This gives a simple way of extending prefix-free codes to product-sets. For example, the set $\mathbb{S}^2 = \mathbb{S} \times \mathbb{S}$ of pairs of strings can be coded by

$$(s_1, s_2) \mapsto \langle s_1 \rangle \langle s_2 \rangle.$$

Or again—an instance we shall apply later—the set $\mathbb{S} \times \mathbb{N}$ can be coded by

$$(s, n) \mapsto \langle s \rangle \langle n \rangle.$$

2.4.5 A second code for $\mathbb{N}$

Definition 2.9. *Suppose $n \in \mathbb{N}$. Then we set*

$$[n] = \langle B(n) \rangle,$$

where $B(n)$ denotes the binary code for n.

Example. Take $n = 6$. Then

$$B(n) = 110,$$

and so

$$[6] = 1111100.$$

Proposition 2.6. *The coding $[n]$ is a maximal prefix-free code for $\mathbb{N}$.*

The conversion from one code for $\mathbb{N}$ to the other is clearly 'algorithmic'. So according to the Church-Turing thesis, there should exist Turing machines S, T that will convert each code into the other:

$$S([n]) = \langle n \rangle, \quad T(\langle n \rangle) = [n].$$

We construct such a machine T in Appendix B. (We leave the construction of S to the reader ....) As we shall see, it may be obvious but it is not simple!

Summary
We have adopted Chaitin's model of a Turing machine. The set $\Omega(T)$ of input strings, or *programs*, for which such a machine T is defined constitutes a *prefix-free* subset of the set $\mathbb{S}$ of all strings.