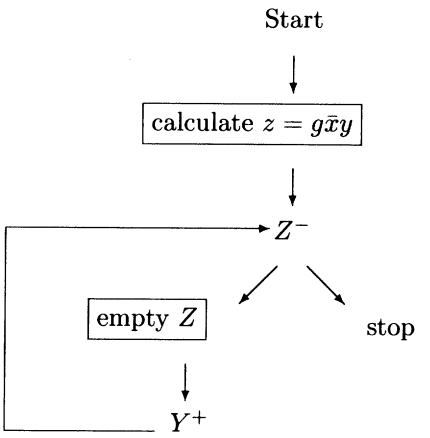

Of course, $f\bar{x}$ is only well-defined if $\forall\bar{x}\exists y(g\bar{x}y = 0)$, otherwise f is not a function but a *partial function.* The class of (partial) functions obtained by adding the minimization scheme to the schemes discussed earlier is the class of *(partial) recursive functions.* These are in fact the only numerical (partial) functions which can be calculated on an abacus or, for that matter, on a Turing machine or on a modern computer.

Exercises

Construct programs for calculating the following numerical functions on an abacus:

1. $z = x + y,\ x \cdot y,\ x^y$;
2. $z = y!$;
3. $y = [x/2]$, where $[\alpha]$ is the greatest integer in α;
4. $y = [\sqrt{x}]$.

20

Recursive and Recursively Enumerable Sets

A set A of natural numbers is said to be *recursive* if there is a recursive (or calculable) function $f : \mathbf{N} \to \mathbf{N}$ such that $A = \{x \in \mathbf{N} | f(x) = 0\}$. In other words, A is recursive provided, for any natural number x, we can determine whether $x \in A$ by performing a calculation f on x. (Actually, it suffices here to take for f a primitive recursive function.) For example, the set of primes is recursive, since we can determine whether a given number is prime by dividing it by all natural numbers less than or equal to its square root. With a bit of work, one can find a recursive function f to do the job.

A set of natural numbers is *recursively enumerable* if there is a recursive function $g : \mathbf{N} \to \mathbf{N}$ such that $A = \{g(0), g(1), g(2), \ldots\}$, where the $g(n)$ are not neccessarily arranged in order of magnitude and may repeat. (Again, it actually suffices to take g to be primitive recursive.) In other words, A is recursively enumerable if there is a calculation which will generate all and only the elements of A in some order, possibly with repetitions. Although the empty set does not conform to the definition above, it will be convenient to consider it to be recursively enumerable.

An important observation linking the above two concepts is the following due to Kleene:

Proposition 20.1. *Let A be a set of natural numbers and A^c its complement in $\mathbf{N}$. Then A is recursive if and only if both A and A^c are recursively enumerable.*

Proof sketched: Suppose A is recursive. Then there is a calculable function f such that $x \in A$ if and only if $f(x) = 0$. We may assume that A is not empty, for otherwise the result holds trivially. Let a be the smallest element

of A. Define $g(x) = x$ if $f(x) = 0$ and $g(x) = a$ if $f(x) \neq 0$. Then g is surely calculable and A is its range. Thus A is recursively enumerable. One shows similarly that A^c is recursively enumerable.

Conversely, suppose both A and A^c are recursively enumerable, both nonempty. Then A is the range of a calculable function g and A^c is the range of a calculable function h. Now every natural number is in the range of g or the range of h and we can determine whether x is an element of A by calculating

$$g(0), h(0), g(1), h(1), g(2), \ldots.$$

We put $f(x) = 0$ if, for some y, $g(y) = x$ and $f(x) = 1$ if, for some y, $h(y) = x$. Then f is calculable and $A = \{x \in \mathbb{N} | f(x) = 0\}$, hence A is a recursive set.

The above considerations can be extended from natural numbers to finite strings of symbols in a formal language, e.g., the language of mathematics. We assume that the set of symbols is finite. For example, if mathematics is based on set theory, we know that the following symbols suffice:

$$x, ' (,), \wedge, \Rightarrow, \vee, \neg, \forall, \exists, =, \in, 0, S.$$

Assuming that there are s symbols in the alphabet (e.g., $s = 14$), we can think of a string of symbols as a natural number expressed in the base s. If you prefer, you may say that the number *encodes* the string.

A mathematical *formula* is a finite string of symbols, combined according to some simple rules, and so is a *proof* in mathematics: a finite list of formulas, each of which is an axiom or else derived from earlier formulas in the list by a rule of inference. There is a finite procedure for 'calculating' whether a formula in a given list meets one of these two requirements, hence whether the given list is a proof. It follows that the set of proofs in mathematics, viewed as the set of natural numbers which encode these proofs, is a recursive set. The last formula in a proof is called a *theorem.*

We can program a computer to look at each natural number in turn, convert it to base s, and determine whether it is (encodes) a formula or a proof. In the second case, we can also tell the computer to print out the number which encodes the last formula of the proof. The computer will then generate a list which consists of all and only those base s natural numbers which encode theorems of mathematics. It follows that the set of theorems in mathematics is recursively enumerable.

On the other hand, it was proved by Alonzo Church that there is no mechanical procedure or algorithm for determining, in general, whether a given formula is a theorem of mathematics. In other words, the set of theorems is not recursive.

If we combine this result of Church with the above proposition of Kleene, we may deduce that the complement of the set of theorems is not recursively enumerable. Moreover, we may deduce the following: