Programming Languages and Computation

Week 9: Computable functions and predicates

* 1. Show that the identity function $\mathrm{id}_{\mathbb{N}} : \mathbb{N} \rightharpoonup \mathbb{N}$ is computable.

Solution

The identity function is computed by the program `skip` (wrt to, say, `x`), because

$$\langle \texttt{skip}, [\mathbf{x} \mapsto n] \rangle \Downarrow [\mathbf{x} \mapsto n]$$

as the relation $\Rightarrow^*$ is reflexive.

* 2. Show that the function $\lfloor \sqrt{-} \rfloor : \mathbb{N} \rightharpoonup \mathbb{N}$ which computes the integer square root of a natural number is computable.

Solution

The integer square root is computed by the following program wrt `n`.

```
i ← 0;
while ((i + 1) * (i + 1) ⩽ n) {      // Invariant: i^2 ⩽ n
  i ← i+1
}
// At this point i^2 ⩽ n < (i+1)^2, so we want to return i.
n ← i; i ← 0
```

* 3. Argue that the variable name does not matter in the definition of computability. That is, if a program `S` computes $f : \mathbb{N} \rightharpoonup \mathbb{N}$ with respect to `x`, then there is a program `S'` that also computes f, but with respect to any variable of our choice.

Solution

The trick is to notice that we can systematically rename all variables in a program to *fresh* ones—i.e. ones that do not already occur in any finite list of variables already in use. Following that, we can rename the 'distinguished' variable `x` to any other variable we please.

** 4. Show that if $f : \mathbb{N} \rightharpoonup \mathbb{N}$ and $g : \mathbb{N} \rightharpoonup \mathbb{N}$ are computable, then so is their composition $g \circ f : \mathbb{N} \rightharpoonup \mathbb{N}$.

Solution

Let `S` be a program that computes f with respect to `x`, i.e. $[\![\texttt{S}]\!]_{\texttt{x}} = f$. Moreover, let `T` be a program that computes g with respect to `x` again. (We can do this because of Q3.) Then, the program `S; T` computes $g \circ f$.

To show this, suppose $(g \circ f)(n) \simeq m$. Then it must be that there is some 'intermediate' natural number n' such that $f(n) \simeq n'$, and $g(n') \simeq m$. As S computes f, we must have

$$\langle \mathtt{S}, [\mathtt{x} \mapsto n] \rangle \Downarrow [\mathtt{x} \mapsto n']$$

Then, we must have that

$$\left\langle \mathtt{T}, [\mathtt{x} \mapsto n'] \right\rangle \Downarrow [\mathtt{x} \mapsto m]$$

Combining these by way of rule (BSeq), it must the case that

$$\langle \mathtt{S}\mathtt{;}\ \mathtt{T}, [\mathtt{x} \mapsto n] \rangle \Downarrow [\mathtt{x} \mapsto m]$$

A similar situation happens if $(g \circ f)(n) \uparrow$: one of the two computations would fail to halt.

Notice that it is very important that the programs S and T 'zero-out' any variables other than x before halting. Otherwise our choice of 'initial' memories $[\mathtt{x} \mapsto n]$ in the definition of computability would not be consistent. For example, T might be written in a way that assumes that some variable's initial value is zero; hence if S had left that variable in a non-zero state, then T might fail to compute the correct result.

* 5. Show that the predicate

$$P = \{n \in \mathbb{N} \mid n \text{ is a prime number}\}$$

is decidable. (NB 1 is *not* a prime number.)

Solution

The following program decides P wrt n.

```
i ← 2; j ← 0;
while (i < n && j = 0) {
  // Try to divide n by i.
  q ← 0; r ← n;
  while (i ⩽ r) { q ← q+1; r ← r-i };
  // If the division is perfect, set a flag that quits the loop.
  if (r = 0) then j ← 1;
  // Try next divisor.
  i ← i+1
}
// If not found a divisor and not 1, return true. Otherwise return false.
if (j = 0 && ! (n = 1)) then n ← 1 else n ← 0;
// Remember to zero out all other variables.
i ← 0; j ← 0; q ← 0; r ← 0
```

** 6. Show that if the predicates P and Q are decidable, then so is $P \cap Q$. Is the same true if P and Q are semi-decidable?

Solution

Let S compute the characteristic function χ_P wrt x, and let T compute the characteristic function wrt y. Then the program

```
x ← z; S;
y ← z; T;
if (x = 1 && y = 1) then z ← 1 else z ← 0;
x ← 0; y ← 0
```

computes $\chi_{P \cap Q}$ wrt z, as long as z is *fresh* (i.e. is not used in either S or T).

The same construction works even if P and Q are only semi-decidable. If z is not in P (resp. Q) then the program given above would run forever: as it is computing the semi-characteristic function, the execution of program S (resp. T) would fail to terminate, and therefore the entire program would fail to terminate.

**** 7. (Trick question.) Is it true that if P and Q are semi-decidable, then so is $P \cup Q$?

Solution

It is in fact true! However, you are not able to show this with the tools you already have. Using the notation of the previous exercise, the key is that the programs S and T must somehow run 'in parallel.' As soon as one of them halts and returns the value 1, then we should halt and return 1 as well. However, the While language is not able to express parallel computation directly.

**** 8. Prove that predicates $U \subseteq \mathbb{N}$ and functions $f : \mathbb{N} \to \mathbb{B}$ to the set $\mathbb{B} = \{\top, \bot\}$ of Boolean values are in a perfect correspondence. That is, prove that there is a bijection between the *powerset* $\mathscr{P}(\mathbb{N}) = \{U \mid U \subseteq \mathbb{N}\}$ and the *function space* $\mathbb{B}^{\mathbb{N}} = \{f \mid f : \mathbb{N} \to \mathbb{B}\}$.

(a) Given a predicate $U \subseteq \mathbb{N}$, construct a function $f_U : \mathbb{N} \to \mathbb{B}$ that corresponds to it.

(b) Given a function $f : \mathbb{N} \to \mathbb{B}$, construct a predicate $U_f \subseteq \mathbb{N}$ that corresponds to it.

(c) Prove that the constructions $U \mapsto f_U$ and $f \mapsto U_f$ are inverses. That is, for a predicate $S \subseteq \mathbb{N}$ prove that $U_{f_S} = S$; and that for a function $h : \mathbb{N} \to \mathbb{B}$ prove that $f_{U_h} = h : \mathbb{N} \to \mathbb{B}$. (What does it mean for two functions to be equal?)