Theory of Computation Excercise for TTF (week 6)

Dimitri Tabatadze · Saturday 06-04-2024

Problem 6.1:

Show that Perdicate $x \mid y$, that is a function $\mid : \mathbb{N}_0 \times \mathbb{N}_0 \to \{\text{true}, \text{false}\}$, given by $x|y = \text{true}$ if and only if x divides y, is primitive recursive.

Solution

We can define

$$x \mid y = \begin{cases} 1 & \text{if } x = y \lor y = 0 \\ 0 & \text{if } (x = 0 \lor y < x) \land y \neq 0 \\ x \mid y - x & \text{if } y > x \land x \neq 0 \end{cases}$$

$$\text{divides}(x, y) = C_{x=y \lor y=0}(x, y) + C_{y>x \land x\neq 0}(x, y) \cdot \text{divides}(x, y \dot{-} x)$$

Every operation and function used for this definition is primitive recursive, so the function itself is also primitive recursive. ■

Problem 6.2:

In the Excercise for Week 5, you showed that a primitive recursive function f, the functions of *bounded sum* and *bounded product* of f, respectively given by

$$\text{bsum}_f(x_0, ..., x_{k-1}, y) = \sum_{i=0}^{y} f(x_0, ..., x_{k-1}, i)$$

$$\text{bprod}_f(x_0, ..., x_{k-1}, y) = \prod_{i=0}^{y} f(x_0, ..., x_{k-1}, i)$$

are also primitive recursive.

Now conclude (by proving) that primitive recursive predicates are closed under *bounded quantification*. That is, show that if P on $\mathbb{N}_0^{k+1}$ is a primitive recursive predicate, then so are the predicates $\forall z \leq y(P(x_0, ..., x_{k-1}, z) = \text{true})$ and $\exists z \leq y(P(x_0, ..., x_{k-1}, z) = \text{true})$.

Solution

- $\forall z \leq y(P(x_0, ..., x_{k-1}, z) = \text{true}) = \text{bforall}_P(x_0, ..., x_{k-1}, y)$
$$\text{bforall}_P(x_0, ..., x_{k-1}, 0) = \text{true}$$
$$\text{bforall}_P(x_0, ..., x_{k-1}, y+1) = C_P(x_0, ..., x_{k-1}, y+1) \cdot \text{bforall}_P(x_0, ..., x_{k-1}, y)$$
- $\exists z \leq y(P(x_0, ..., x_{k-1}, z) = \text{true}) = \text{bexists}_P(x_0, ..., x_{k-1}, y)$
$$\text{bexists}_P(x_0, ..., x_{k-1}, 0) = \text{false}$$
$$\text{bexists}_P(x_0, ..., x_{k-1}, y+1) = \text{sg}(C_P e(x_0, ..., x_{k-1}, y+1) + \text{bexists}_P(x_0, ..., x_{k-1}, y))$$

■

Problem 6.3: *

In the lecture we sketched a proof that not every computable function is primitive recursive.

a) Where does this proof fail for μ-recursive functions?
b) Where does this proof fail if we only consider total (defined everywhere) μ-recursive functions?

Note that the set of μ-recursive functions, as well as its proper subset of total μ-recursive functions are countably infinite.

Solution

a) You can't add 1 to Ω.
b) It's imposible to create a parser that classifies total μ-recursive functions since if a function is undefined at some point, it will take infinite time to know that.

■

Problem 6.4:

What follows are the exercises from the lecture on Turing Machine. See the mentioned lecture for precise definitions.
Construct a Turing machine for

a) "decrementing binary numbers." That is, given an input of a binary number $\text{bin}(n)$, it gives an output $\text{bin}(n-1)$;
b) "concatenating tape inscriptions." That is two machines, one giving `tape1 = tape1#tape2`. and the other `tape1 = tape2#tape1`.
c) "head and tail of tapes." That machines giving `tape2 = head(tape1)` and `tape2 = tail(tape1)`.

Solution

For all of the machines z_e is the end state and z_0 is the initial state

a) `tape1 = bin(int(tape1) - 1)`

$$\begin{aligned} \delta(z_0, a) &= (z_0, a, R) & \delta(z_0, B) &= (q_0, B, L) \\ \delta(q_0, 0) &= (q_0, 1, L) & \delta(q_0, 1) &= (q_1, 0, L) \\ \delta(q_1, a) &= (q_1, a, L) & \delta(q_1, B) &= (q_2, B, R) \\ \delta(q_2, 0) &= (q_2, B, R) & \delta(q_2, 1) &= (z_e, 1, N) \\ \delta(q_2, B) &= (z_e, 0, N) \end{aligned}$$

b) • `tape1 = tape1#tape2`

$$\begin{aligned} \delta(z_0, a_1, a_2) &= (z_0, a_1, a_2, R, N) & \delta(z_0, B, a_2) &= (z_1, \#, B, R, N) \\ \delta(z_1, B, a_2) &= (z_1, a_2, B, R, R) & \delta(z_1, B, B) &= (z_2, B, B, L, N) \\ \delta(z_2, a_1, B) &= (z_2, a_1, B, L, N) & \delta(z_2, B, B) &= (z_e, B, B, R, N) \end{aligned}$$

• `tape1 = tape2#tape1`

$$\delta(z_0, a_1, a_2) = (z_0, a_1, a_2, N, R) \qquad \delta(z_0, a_1, B) = (z_1, a_1, \#, R, R)$$
$$\delta(z_1, a_1, B) = (z_1, B, a_1, R, R) \qquad \delta(z_1, B, B) = (z_2, B, B, L, N)$$
$$\delta(z_2, B, a_2) = (z_2, a_2, B, L, N) \qquad \delta(z_2, B, B) = (z_e, B, B, R, N)$$

c)
- `tape2 = head(tape1)`

$$\delta(z_0, a_1, a_2) = (z_0, a_1, B, N, R) \qquad \delta(z_0, a_1, B) = (z_1, B, a_1, R, N)$$
$$\delta(z_1, a_1, a_2) = (z_1, B, a_2, R, N) \qquad \delta(z_1, B, a_2) = (z_e, B, a_2, N, N)$$

- `tape2 = tail(tape1)`

$$\delta(z_0, a_1, a_2) = (z_0, a_1, B, R, R) \qquad \delta(z_0, a_1, B) = (z_0, a_1, B, R, N)$$
$$\delta(z_0, B, B) = (z_1, B, B, L, N) \qquad \delta(z_1, a_1, B) = (z_2, B, a_1, L, N)$$
$$\delta(z_2, a_1, a_2) = (z_2, B, a_2, L, N) \qquad \delta(z_2, B, a_2) = (z_e, B, a_2, N, N)$$

■