Fundamentele Informatica 3

Solutions to selected exercises from Chapter 10
John Martin: Introduction to Languages and the Theory of Computation
(fourth edition)

Rudy van Vliet

Spring 2014

10.1 The set F includes the set of total functions from $\mathbb{N}$ to $\mathbb{N}$, which includes the set of total functions from $\mathbb{N}$ to $\{0, 1\}$, which in turn is the same size as $2^{\mathbb{N}}$, which is uncountable (see Example 8.31 from the book). This implies that F is certainly uncountable.

By definition, each computable (partial) function can be computed by a Turing machine, and each Turing machine computes at most one function. Therefore, C, the set of computable, partial functions is at most as large as the set of Turing machines. By Example 8.30 from the book, the latter set is countable. This implies that C is certainly countable.

Now, by Exercise 8.38 from the book, $U = F - C$ is uncountable.

10.2 A solution to this exercise is given at page 419 of the book.

10.14 We have

$$\begin{aligned} f(k+1) &= (k+1)^2 = k^2 + 2k + 1 = f(k) + 2k + 1 \\ f(0) &= 0 \end{aligned}$$

Therefore, take

$$\begin{aligned} g(\cdot) &= 0 = C_0^0(\cdot) \\ h(x_1, x_2) &= x_2 + 2x_1 + 1 \end{aligned}$$

Then, indeed,

$$\begin{aligned} f(0) &= g(\cdot) \\ f(k+1) &= h(k, f(k)) \end{aligned}$$

10.15 a Various solutions are possible. We give two.
(1)

$$f(x,y) = \mathit{Add}(\mathit{Mult}(C_2^2(x,y), p_1^2(x,y)), \mathit{Mult}(C_3^2(x,y), p_2^2(x,y)))$$

with functions *Add* and *Mult* from Example 10.5 from the book.
(2)

$$f(x,y) = \mathit{Add}(\mathit{Double}(p_1^2(x,y)), \mathit{Triple}(p_2^2(x,y)))$$

where the function *Double* is defined by

$$\begin{aligned} \mathit{Double}(0) &= 2 \times 0 = 0 \\ \mathit{Double}(k+1) &= 2 \times (k+1) = 2 \times k + 2 = \mathit{Double}(k) + 2 = s(s(\mathit{Double}(k))) \end{aligned}$$

In other words, *Double* is obtained by primitive recursion from the functions

$$\begin{aligned} g(\cdot) &= C_0^0(\cdot) \\ h(x_1, x_2) &= x_2 + 2 = s(s(p_2^2(x_1, x_2))) \end{aligned}$$

The function *Triple* is defined analogously.
c We have

$$\begin{aligned} f(0) &= 2^0 = 1 \\ f(k+1) &= 2^{k+1} = 2 \times 2^k = 2 \times f(k) \end{aligned}$$

Therefore, take

$$\begin{aligned} g(\cdot) &= C_1^0(\cdot) \\ h(x_1, x_2) &= 2 \times x_2 = \mathit{Mult}(C_2^2(x_1, x_2), p_2^2(x_1, x_2)) \end{aligned}$$

Then f is obtained from the functions g and h by primitive recursion.

10.19 b We have

$$f(x,y) = \begin{cases} x & \text{if } x \leq y \\ y & \text{if } x > y \end{cases}$$

Indeed, the functions

$$\begin{aligned} f_1(x,y) &= x = p_1^2(x,y) \\ f_2(x,y) &= y = p_2^2(x,y) \end{aligned}$$

are primitive recursive, by Theorem 10.6 from the book the two-place predicates $x \leq y$ and $x > y$ are also primitive recursive, and for every $(x,y) \in \mathbb{N}^2$,

exactly one of the two predicates is true. Therefore, by Theorem 10.7 from the book, f is primitive recursive.
c We have

$$f(x) = \max\{y \mid y^2 \leq x\}$$

which can be reformulated as

$$f(x) = \min\{y \mid y^2 > x\} \dot{-} 1 \tag{1}$$

Let the two-place predicate P be defined by $P(x, y) = (y^2 > x)$. Because the function $f_1(y) = y^2 = \mathit{Mult}(p_1^1(y), p_1^1(y))$ is primitive recursive, and by Theorem 10.6 from the book the two-place predicate $x_1 > x_2$ is primitive recursive, P is also primitive recursive.

In order to prove that f is primitive recursive, we have to bound the possible values for y in Equation (1), without affecting the validity of the equation. We can take as bound $k = x + 1$, because for sure,

$$(x + 1)^2 = x^2 + 2x + 1 > x$$

We thus have

$$f(x) = m_P(x, x + 1) \dot{-} 1$$

By Theorem 10.12 from the book, m_P is primitive recursive, and so is f.

10.22 The function *HighestPrime* : $\mathbb{N} \to \mathbb{N}$ is defined by

$$\mathit{HighestPrime}(k) = \begin{cases} 0 & \text{if } k \leq 1 \\ \max\{i \mid \mathit{Exponent}(i, k) > 0\} & \text{if } k \geq 2 \end{cases} \tag{2}$$

Because the function *Exponent* is primitive recursive, so is the predicate P defined by

$$P(k, i) = (\mathit{Exponent}(i, k) > 0)$$

But then, by Exercise 10.23, the bounded maximalisation m^P of P, defined by

$$m^P(k, k_2) = \begin{cases} \max\{i \leq k_2 \mid \mathit{Exponent}(i, k) > 0\} & \text{if this set is not empty} \\ 0 & \text{otherwise} \end{cases}$$

is also primitive recursive.

If we can find a bound k_2 such that for $k \geq 2$,

$$\max\{i \mid \mathit{Exponent}(i, k) > 0\} = \max\{i \leq k_2 \mid \mathit{Exponent}(i, k) > 0\}$$

then *HighestPrime* is primitive recursive, because then Equation (2) reduces to

$$\mathit{HighestPrime}(k) = \begin{cases} 0 & \text{if } k \leq 1 \\ m^P(k, k_2) & \text{if } k \geq 2 \end{cases} \tag{3}$$

Note that indeed, for $k \geq 2$, the set $\{i \mid \mathit{Exponent}(i,k) > 0\}$ is not empty.

For this, we observe that $\mathit{Exponent}(i,k) > 0$, if and only if $\mathit{PrNo}(i)$ divides k evenly. If this is the case, then certainly $\mathit{PrNo}(i) \leq k$. If that is the case, then certainly $i \leq k$, because for every $i \geq 0$, $i \leq \mathit{PrNo}(i)$. Therefore, we can safely bound the values for i by $k_2 = k$:

$$\begin{aligned} \mathit{HighestPrime}(k) &= \begin{cases} 0 & \text{if } k \leq 1 \\ \max\{i \leq k \mid \mathit{Exponent}(i,k) > 0\} & \text{if } k \geq 2 \end{cases} \\ &= \begin{cases} 0 & \text{if } k \leq 1 \\ m^P(k,k) & \text{if } k \geq 2 \end{cases} \end{aligned}$$

10.23 A solution to this exercise is given at page 420 of the book.
b *Alternative solution:* Assume that the two-place predicate P is primitive recursive. The given definition of m^P is equivalent to

$$\begin{aligned} m^P(X,k) &= \begin{cases} k \dot{-} \min\{y \leq k \mid \ P(X, k \dot{-} y) \text{ is true}\} & \text{if this set is not empty} \\ 0 & \text{otherwise} \end{cases} \\ &= k \dot{-} \begin{cases} \min\{y \leq k \mid \ P(X, k \dot{-} y) \text{ is true}\} & \text{if this set is not empty} \\ k+1 & \text{otherwise} \end{cases} \end{aligned}$$

If we can prove that the function P' defined by

$$P'(x,k) = \begin{cases} \min\{y \leq k \mid \ P(X, k \dot{-} y) \text{ is true}\} & \text{if this set is not empty} \\ k+1 & \text{otherwise} \end{cases}$$

is primitive recursive, then so is m^P.

For this, let Q be the the three-place predicate defined by

$$Q(x,k,y) = P(x, k \dot{-} y).$$

Because P is primitive recursive, so is Q. By Theorem 10.12 from the book, its bounded minimalization m_Q defined by

$$m_Q(x,k,k_2) = \begin{cases} \min\{y \leq k_2 \mid \ P(X, k \dot{-} y) \text{ is true}\} & \text{if this set is not empty} \\ k_2+1 & \text{otherwise} \end{cases}$$

is primitive recursive. But then our function P' is also primitive recursive, because

$$P'(x,k) = m_Q(x,k,k).$$

10.35 We ignore the suggestion in the book, because it would make things more complicated.

As a result of a move of the Turing machine for configuration number m, the symbol at position $\mathit{Posn}(m)$ on the tape changes from $\mathit{Symbol}(m)$ to $\mathit{NewSymbol}(m)$. The rest of the tape remains the same in this move.

Position $\mathit{Posn}(m)$ contributes a factor $\mathit{PrNo}(\mathit{Posn}(m))^{\mathit{Symbol}(m)}$ to $\mathit{TapeNumber}(m)$. In $\mathit{NewTapeNumber}(m)$, this contribution is substituted by a factor $\mathit{PrNo}(\mathit{Posn}(m))^{\mathit{NewSymbol}(m)}$. We can thus write

$$\mathit{NewTapeNumber}(m) = (\mathit{TapeNumber}(m)\ \mathit{Div}\ \mathit{PrNo}(\mathit{Posn}(m))^{\mathit{Symbol}(m)}) \times \mathit{PrNo}(\mathit{Posn}(m))^{\mathit{NewSymbol}(m)}.$$

---

version 3 June 2014