Solutions to Selected Exercises of *Theoretical Computer Science for the Working Category Theorist*

Noson S. Yanofsky*

October 27, 2019

Feel free to email me more solutions. I will add other solutions to this document (with your name attached to it.)

Exercise 2.1.3. For any object $f\colon b \longrightarrow a$ of $\mathbb{A}/a$, f is the unique morphism to the identity that makes the triangle commute.

Exercise 2.1.7. This is similar to the solution to Exercise 2.1.3.

Exercise 3.1.2.

- $Length\colon String \longrightarrow Nat.$
- $Desc\colon Nat \longrightarrow String.$
- $Lines\colon String \longrightarrow Nat.$

Exercise 3.1.4.

- $MaxMeanMin\colon Nat^* \longrightarrow Nat \times Real \times Nat.$

*Department of Computer and Information Science, Brooklyn College CUNY, Brooklyn, N.Y. 11210. And Computer Science Department, The Graduate Center CUNY, New York, N.Y. 10016. e-mail: noson@sci.brooklyn.cuny.edu

- $Eval : Nat^{Nat} \times Nat \longrightarrow Nat$.
- $LastTwinPrimes : Nat \times Nat \longrightarrow Bool$. The interesting part of this problem is that it is unknown if there is a last twin prime or if there are an infinite number of them.

Exercise 3.1.7. The terminal type $*$ because $T \times * = T = * \times T$.

Exercise 3.1.11.

- Matrix multiplication is a totally computable function.
- We will see in Example 4.2.3 that this is not in $\mathbb{TotCompFunc}$. One can gain an intuition about this from the following. We might try a few inputs to the program and see if any of them halt. If they do, then we know the domain is not empty. But how are we to know if the domain is totally empty? We would have to go through an infinite number of possible inputs. This shows that the obvious way of determining if the domain of a function is empty does not work. Perhaps there is a cleverer way of determining this function. We will mathematically prove later that no such clever method exists.
- This function is totally computable. It is defined as: $x \dot{-} y = Max\{x-y, 0\}$.

Exercise 3.2.6.

(i) The word is presented as a single string of a's and b's. First the Turing machine goes through the entire input to make sure that all the a's are before the b's. If not, reject. If it is in the right form, start at the beginning of the input and for every a, add one to a counter on a work tape. When you get to the b's, decrement the counter. At the end of the input, if the counter is zero, accept the word, otherwise, reject the word.

(ii) The number n is on the input tape. Use a standard method to calculate $\lfloor \sqrt{n} \rfloor$. Place the result on a work tape. Then systematically go from $m = 2$ till $m = \lfloor \sqrt{n} \rfloor$. If any m evenly divides into n reject the input as not prime, otherwise accept as prime.

(iii) $n^m : Nat \times Nat \longrightarrow Nat$ can be done by performing a loop m times and multiplying by n in every loop (we learned how to multiply in Example 3.2.5.) In detail, use two work tapes. One tape will be a counter for the loop and the other will have the product. The computer then performs the following program.

(a) Set counter tape to m

(b) Set product tape to 1

(c) If counter$=0$ goto step 7

(d) Multiply n times the product

(e) Decrement the counter

(f) Goto Step 3

(g) Transfer the product to the output tape

Exercise 3.2.7. Assume that T_1 has t_1 tapes and T_2 has t_2 tapes. Assume Q_1 is the set of states of T_1 and Q_2 is the set of states of T_2. The tensor of the two machines has $Q_1 \times Q_2$ as the set of states. The transition function will be given as follows:

$$\delta((q_i, q_j); x_1, x_2, \ldots x_{t_1+t_2}) = ((q_{i'}, q_{j'}); y_1, y_2, \ldots, y_{t_1+t_2}; D_1, D_2, \ldots, D_{t_1+t_2})$$

if and only if

$$\delta_1(q_i; x_1, x_2, \ldots x_{t_1}) = (q_{i'}; y_1, y_2, \ldots, y_{t_1}; D_1, D_2, \ldots, D_{t_1})$$

and

$$\delta_2(q_j; x_{t_1+1}, x_{t_1+2}, \ldots x_{t_1+t_2}) = (q_{j'}; y_{t_1+1}, y_{t_1+2}, \ldots, y_{t_1+t_2}; D_{t_1+1}, D_{t_1+2}, \ldots, D_{t_1+t_2})$$

where $D_i \in \{L, R\}$.

Exercise 3.2.10. In general, for every function in $\mathbb{C}\mathbb{o}\mathbb{m}\mathbb{p}\mathbb{S}\mathbb{t}\mathbb{r}\mathbb{i}\mathbb{n}\mathbb{g}$, there are many Turing machines/programs that can compute it.

Exercise 3.3.7.

Zero Function	Successor Function	Projector Functions
1. $Y_1 = Y_1 - 1$ 2. If $Y_1 \neq 0$ Goto 1	1. $Y_1 = X_1$ 2. $Y_1 = Y_1 + 1$	1. $Y_1 = X_i$

Exercise 3.3.8.

Composition	Recursion
Assume that $f_1, f_2, \ldots, f_n$ and g are computed by programs $F_1, F_2, \ldots, F_n$, and G, then the following program will compute function h. 1. $W_1 = F_1(X_1, X_2, \ldots, X_m)$ 2. $W_2 = F_2(X_1, X_2, \ldots, X_m)$ 3. $\vdots$ 4. $W_n = F_n(X_1, X_2, \ldots, X_m)$ 5. $Y_1 = G(W_1, W_2, \ldots, W_n)$	Assume f and g are computed by programs F and G, then the following program will compute function h. 1. $Y_1 = F(X_1, X_2, \ldots, X_m)$ 2. If $X_{n+1} = 0$ goto 6 3. $Y_1 = G(X_1, \ldots, X_m, Y_1)$ 4. $X_{n+1} = X_{n+1} - 1$ 5. Goto 2 6. Stop.

and

μ-minimization
1. $Y_1 = 0$ 2. $W_1 = F(X_1, ..., X_m, Y_1)$ 3. If $W_1 = 0$ goto 6 4. $Y_1 = Y_1 + 1$ 5. Goto 2 6. Stop

Exercise 3.3.11.

Sign	Distance
$sg(0) = 0$ $sg(s(x)) = 1.$	$\lvert x - y \rvert = (x - y) + (y - x).$

Remainder
$rem(x, 0) = 0$ $rem(x, s(y)) = (rem((x, y) + 1) * (sg(\lvert x - (rem(x, y) + 1) \rvert).$

Exercise 3.3.14. $f^{-1}(x) = \mu_y[f(y) = x].$

Exercise 3.5.7. This falls out of the definition of equivalence. Two $\{\Psi\}_n^m$ are logically equivalent if they describe the same computable function.

Exercise 4.3.1. Send the input to the query tape and then call the oracle.

Exercise 4.3.2. Rather than using the oracle, we can just put in a computable subroutine that computes the function.

Exercise 4.3.3. Let T be the machine that computes f and uses the g oracle. Let T' be the machine that computes g using the h oracle. Make a new machine T'' that computes f as follows: T'' should be like T, however, rather than query the g oracle, it goes into a whole module that does what T' does including call h.

Exercise 5.2.8. To go from the Subset sum problem to the knapsack problem we take the given $\{s_1, s_2, \ldots, s_n\}$ and make the sizes and the profits equal those numbers. We also set the $G = C$.