Exercise sheet 6: Review – Corrections

Turing machines

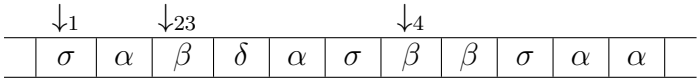
1. Consider a k-head Turing machine having a single tape and k heads; more than one head can be on the same cell at a time. At each move, the TM will read the symbols under its heads, and consider its internal state (a unique state for whole machine), then it changes the state, writes a symbol on each cell under a head (if there are more than one head in the same cell, it writes the symbol with only one of them) and moves each head to the left or to the right independently. Prove that the languages accepted by k-head Turing machines are the same languages accepted by ordinary TM's.

   Solution We must prove that the k head TM can be simulated by an ordinary TM. Since we already know that a multiple-tape TM can be simulated with an ordinary one, it suffices to simulate the k-head TM with a multiple-tape TM. We call K to the k-tape TM and M to the multiple-tape TM that simulates the former.

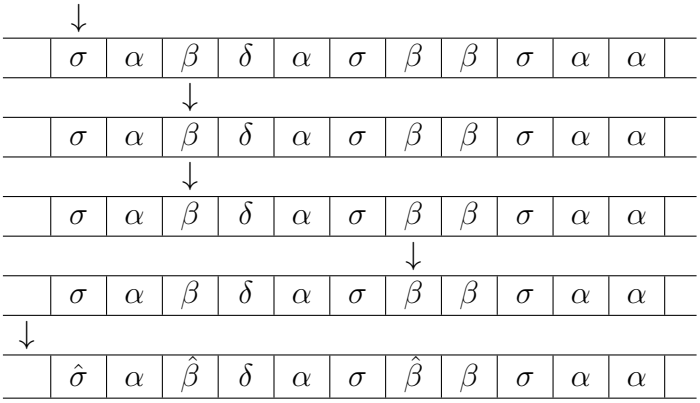
   Assume we enumerate each head as h_i with $i = 1, \ldots, k$. We use a $k+1$-tape TM M to do the simulation. The first step is to copy the input tape of K into each tape of M. In the last tape, we call it 'extra', we add a mark to each symbol where there is a head in K (add a mark means that the alphabet of machine M is the union of the alphabet of the machine K plus the set of all the symbols in the alphabet, with a mark).

   So, we get the following picture:

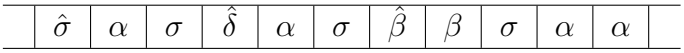
   Let the machine K be in the following configuration:

   Then the machine M is in the following configuration:

   At each move in machine K, we do the same movement in the corresponding tape. Then, we check the last (extra) tape to see what are the changes with respect to each tape. We synchronize those changes one by one in the extra tape. For example, if in the tape 2th the 3th cell has been changed from β to σ and the head moves to the right, then after synchronizing such tape, the extra tape will looks as follows

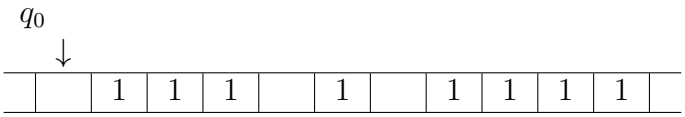
   Then when we have to synchronize the 3th tape we compare all the cells except those already synchronized (all except the 3th cell in our example), so the extra tape remain unchanged.

   After having synchronized all the tapes. We replicate the extra tape in all the other tapes, which finish the synchronization process and the movement.

2. Consider the numeric functions $f : \mathbb{N}_0^k \to \mathbb{N}_0$.

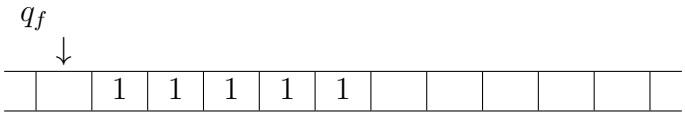
   - Natural numbers can be represented in unary form: *e.g.* $\bar{0} = 1$, $\bar{1} = 11$, $\bar{2} = 111$, ... Hence we can take the input alphabet of a TM computing numeric functions to be $\Sigma = \{1\}$.
   - Consecutive numbers on the input are separated by a blank space $\square$.
   - The TM starts its computation with the head placed on the $\square$ preceding the first number.

     e.g. for $f(2,0,3)$:



   - When the computation terminates, then input has been replaced on the tape with the result of the function.

     e.g. if $f(2,0,3) = 4$:



   We make some assumptions to ease composition of TM's:

   - There is a single final state q_f. The TM halts with the head over the $\square$ just before the solution.
   - The only transition from q_0 is $\delta(q_0, \square) = (q_i, \square, R)$.
   - There are no transitions entering q_0 or of the form $\delta(q_f, \square)$.
   - The computation loops whenever $f(m)$ is undefined.

   This allows us to sequentially compose TM's, as represented by the following diagram

$$\xrightarrow{\text{(I)}} \boxed{M_1} \xrightarrow{\text{(II)}} \boxed{M_2} \xrightarrow{\text{(III)}}$$

   I Initial state of M_1 and of the combination.

   II Final state of M_1 and initial state of M_2.

   III Final state of M_2 and also of the combination.

   (a) Construct a TM computing the successor function $s(n) = n + 1$.
   SOLUTION

$$\begin{aligned} \delta(q_0, \square) &= (q_1, \square, R) \\ \delta(q_1, 1) &= (q_1, 1, L) \\ \delta(q_1, \square) &= (q_f, 1, L) \end{aligned}$$

   (b) Construct a TM computing the zero function $z(X^k) = 0$.
   SOLUTION

$$\begin{aligned} \delta(q_0, \square) &= (q_1, \square, R) \\ \delta(q_1, 1) &= (q_1, \square, R) \\ \delta(q_1, \square) &= (q_2, \square, R) \\ \delta(q_2, 1) &= (q_1, \square, R) \\ \delta(q_2, \square) &= (q_f, 1, L) \end{aligned}$$

   (c) Construct a TM computing the empty function, *i.e.* the function that is undefined for every $n \in \mathbb{N}_0$.
   SOLUTION

$$\begin{aligned} \delta(q_0, \square) &= (q_1, \square, R) \\ \delta(q_1, 1) &= (q_1, 1, R) \\ \delta(q_1, \square) &= (q_1, \square, R) \end{aligned}$$

(d) Construct a TM computing the projector $u_k^{(n)}(x_1, \ldots, x_k, \ldots, x_n) = x_k$.
SOLUTION

$$\begin{aligned}
\delta(q_0, \square) &= (q_1, \square, R) \\
\delta(q_i, 1) &= (q_i, \square, R) \quad \text{with } i = 1, \ldots, k-1 \\
\delta(q_i, \square) &= (q_{i+1}, \square, R) \quad \text{with } i = 1, \ldots, k-1 \\
\delta(q_k, 1) &= (q_k, 1, R) \\
\delta(q_k, \square) &= (q_m, \square, R) \\
\delta(q_m, 1) &= \delta(q_m, \square, R) \\
\delta(q_m, \square) &= (q_n, \square, R) \\
\delta(q_n, 1) &= (q_m, \square, R) \\
\delta(q_n, \square) &= (q_v, \square, L) \\
\delta(q_v, \square) &= (q_v, \square, L) \\
\delta(q_v, 1) &= (q_w, 1, L) \\
\delta(q_w, 1) &= (q_w, 1, L) \\
\delta(q_w, \square) &= (q_l, \square, L) \\
\delta(q_l, \square) &= (q_f, \square, R)
\end{aligned}$$

(e) Construct a TM computing the predecesor function $Pd(n) = \begin{cases} 0 & \text{if } n = 0 \\ n-1 & \text{otherwise} \end{cases}$
SOLUTION

$$\begin{aligned}
\delta(q_0, \square) &= (q_1, \square, R) \\
\delta(q_1, 1) &= (q_2, \square, R) \\
\delta(q_2, \square) &= (q_f, 1, R) \\
\delta(q_2, 1) &= (q_f, 1, L)
\end{aligned}$$

(f) Using sequential composition, construct a TM computing the constant function *one*. Tip: $one = \Phi(s^{(1)}, z^{(n)})$.
SOLUTION

$$z^{(n)} \begin{cases} \delta(q_0, \square) = (q_1, \square, R) \\ \delta(q_1, 1) = (q_1, \square, R) \\ \delta(q_1, \square) = (q_2, \square, R) \\ \delta(q_2, 1) = (q_1, \square, R) \\ \delta(q_2, \square) = (q_{\mathbf{3}}, 1, L) \end{cases}$$

$$s^{(1)} \begin{cases} \delta(q_{\mathbf{3}}, \square) = (q_{\mathbf{4}}, \square, R) \\ \delta(q_{\mathbf{4}}, 1) = (q_{\mathbf{4}}, 1, L) \\ \delta(q_{\mathbf{4}}, \square) = (q_f, 1, L) \end{cases}$$

Recursive functions

3. Let $\lessdot$ be a primitive recursive relation. Show that the following functions are recursive.

(a) $f_1(x, y_0, y) =$ the first value z in $[y_0, y]$ for which $x \lessdot z$.
SOLUTION Notice that

$$f_1(x, y_0, y) = \mu_z(^\circ D(\chi_{\geq}(z, y_0).\chi_{\geq}(y, z).\chi_{\lessdot}(x, z)) = 0)$$

so let $g(z, x, y_0, y)$ be the following function

$$\Phi(^\circ D, \Phi(\Pi, \Phi(\Pi, \Phi(\chi_{\geq}, u_1^{(4)}, u_3^{(4)}), \Phi(\chi_{\geq}, u_4^{(4)}, u_1^{(4)})), \Phi(\chi_{\lessdot}, u_2^{(4)}, u_1^{(4)})))(z, x, y_0, y)$$

Then $f_1 = M[g]$.

(b) $f_2(x,y)$ = the second value z in $[0,y]$ for which $x \lessdot z$.
SOLUTION Notice that

$$f_2(x,y) = f_1(x, f_1(x,0,y)+1, y)$$

So $f_2 = \Phi(f_1, u_1^{(2)}, \Phi(f_1, u_1^{(2)}, z^{(2)}, u_2^{(2)}), u_2^{(2)})$.

4. Let f and g be primitive recursive functions. Show that the following function is also primitive recursive:

$$h(x) = \begin{cases} 1 & \text{if } f(i) > g(j), \text{ for all } 0 \le i \le x \text{ and } 0 \le j \le x \\ 0 & \text{otherwise} \end{cases}$$

SOLUTION We define h using recursion.

$$\begin{aligned} h(0) &= \begin{cases} 1 & \text{if } f(i) > g(j), \text{ for all } 0 \le i \le 0 \text{ and } 0 \le j \le 0 \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } f(0) > g(0) \\ 0 & \text{otherwise} \end{cases} \\ &= \chi_{>}(f(0), g(0)) = \Phi(\chi_{>}, \Phi(f, z^{(0)}), \Phi(g, z^{(0)})) \\ h(x+1) &= h(x).\chi_{>}(f(x+1), g(x+1)) \\ &= \Phi(\Pi, u_2^{(2)}, \Phi(\chi_{>}, \Phi(f, \Phi(s, u_1^{(2)})), \Phi(g, \Phi(s, u_1^{(2)}))))(x, h(x)) \end{aligned}$$

So $h = R(\Phi(\chi_{>}, \Phi(f, z^{(0)}), \Phi(g, z^{(0)})), \Phi(\Pi, u_2^{(2)}, \Phi(\chi_{>}, \Phi(f, \Phi(s, u_1^{(2)})), \Phi(g, \Phi(s, u_1^{(2)})))))$.

5. Give primitive recursive definitions for the following functions

(a) $\text{half}(x) = \lfloor \frac{x}{2} \rfloor$.
SOLUTION We define half using recursion.

$$\begin{aligned} \text{half}(0) &= 0 = z^{(0)} \\ \text{half}(x+1) &= \text{half}(x) + \chi_{odd}(x) \\ &= \Phi(\Sigma, u_2^{(2)}, \Phi(\chi_{odd}, u_1^{(2)}))(x, \text{half}(x)) \end{aligned}$$

So half$= R(z^{(0)}, \Phi(\Sigma, u_2^{(2)}, \Phi(\chi_{odd}, u_1^{(2)})))$.

(b) $\min(x,y)$.
SOLUTION Notice that $\min(x,y) = y \dot{-} (y \dot{-} x)$: When $x \ge y$, $y \dot{-} x = 0$ so this expression is 0. When $x < y$, $y \dot{-} (y \dot{-} x) = y - (y - x) = y - y + x = x$. Then

$$\min = \Phi(\dot{-}d, u_2^{(2)}, \Phi(\dot{-}d, u_2^{(2)}, u_1^{(2)}))$$

(c) $\min^n(x_1, \ldots, x_n)$ for all $n \ge 2$.
SOLUTION We proceed by recursion.

$$\min^n(x_1, \ldots, 0) = 0 = z^{(n-1)}(x_1, \ldots, x_{n-1})$$

$$\min^n(x_1, \ldots, x_n+1) = \begin{cases} \min^n(x_1, \ldots, x_n) & \text{if } \min^n(x_1, \ldots, x_n) \ne x_n \\ x_n + 1 & \text{if } \min^n(x_1, \ldots, x_n) = x_n \\ & \wedge\, \forall i \in \{1, \ldots, n-1\},\ x_i \ne x_n \\ x_n & \text{if } \min^n(x_1, \ldots, x_n) = x_n \\ & \wedge\, \exists i \in \{1, \ldots, n-1\},\ x_i = x_n \end{cases}$$

$$= {}^{\circ}D(E(u_{n+1}^{(n+1)}, u_n^{(n+1)})).u_{n+1}^{(n+1)} + E(u_{n+1}^{(n+1)}, u_n^{(n+1)}).(u_n^{(n+1)} + \prod_{i=1}^{n-1} {}^{\circ}D(E(u_i^{(n+1)}, u_n^{(n+1)})))(X^n, \min^n(X^n))$$

$$= \underbrace{\Phi(\Sigma, \Phi(\Pi, \Phi({}^{\circ}D, \Phi(E, u_{n+1}^{(n+1)}, u_n^{(n+1)})), u_{n+1}^{(n+1)}), \Phi(\Pi, \Phi(E, u_{n+1}^{(n+1)}, u_n^{(n+1)}), \Phi(\Sigma, u_n^{(n+1)}, g)))}_{h}(X^n, \min^n(X^n))$$

where $X^n = (x_1, \ldots, x_n)$ and g is defined as follows

$$g^{(n+1)}(X) = \prod_{i=1}^{n-1} {}^{\circ}D(E(u_i^{(n+1)}, u_n^{(n+1)}))(X) = \Phi(\Pi, f_1, \Phi(\Pi, f_2, \ldots, \Phi(\Pi, f_{n-2}, f_{n-1})))(X)$$

where $\forall i = 1, \ldots, n-1$, $f_i = {}^{\circ}D(E(u_i^{(n+1)}, u_n^{(n+1)}))$. Notice that g is a well defined primitive recursive function since n is fixed.
Then $\min^n = R(z^{(n-1)}, h)$.

(d) $\max(x, y)$.
SOLUTION Notice that $\max(x, y) = x + (y \dot{-} x)$: When $x \geq y$, $y \dot{-} x = 0$ so this expression is x. When $x < y$, $x + (y \dot{-} x) = x + (y - x) = y$. Then

$$\max = \Phi(\Sigma, u_1^{(2)}, \Phi({}^{\circ}d, u_2^{(2)}, u_1^{(2)}))$$

(e) $rem(a, b)$ = remainder of the division of a by b (consider $rem(a, 0) = a$).
SOLUTION We proceed by recursion, however we will do a recursion over a instead of a recursion over b. Since the operator R is defined for recursion over the last argument, we first define a function $rm(b, a)$ = remainder of the division of a by b, and then we define rem in terms of rm.

$$\begin{aligned} rm(b, 0) &= 0 = z^{(1)}(b) \\ rm(b, a+1) &= (rm(b, a) + 1).{}^{\circ}D(E(rm(b, a) + 1, b)) \\ &= \Phi(\Pi, \Phi(s, u_3^{(3)}), \Phi({}^{\circ}D, \Phi(E, \Phi(s, u_3^{(3)}), u_1^{(3)})))(b, a, rm(b, a)) \end{aligned}$$

Then $rm = R(z^{(1)}, \Phi(\Pi, \Phi(s, u_3^{(3)}), \Phi({}^{\circ}D, \Phi(E, \Phi(s, u_3^{(3)}), u_1^{(3)}))))$
Notice that $rem(a, b) = rm(b, a)$, then $rem = \Phi(rm, u_2^{(2)}, u_1^{(1)})$.

(f) $quo(a, b)$ = quotient of the division of a by b, with $quo(a, 0) = 0$.
SOLUTION Again we want the recursion on the first argument, so we define $q(b, a)$ as the quotient of the division of a by b.

$$\begin{aligned} q(b, 0) &= 0 = z^{(1)}(b) \\ q(b, a+1) &= q(b, a) + {}^{\circ}D(rem(a+1, b)) \\ &= \Phi(\Sigma, u_3^{(3)}, \Phi({}^{\circ}D, \Phi(rem, \Phi(s, u_2^{(3)}), u_1^{(3)})))(b, a, q(b, a)) \end{aligned}$$

Then $q = R(z^{(1)}, \Phi(\Sigma, u_3^{(3)}, \Phi({}^{\circ}D, \Phi(rem, \Phi(s, u_2^{(3)}), u_1^{(3)}))))$ and since $quo(a, b) = q(b, a)$, $quo = \Phi(q, u_2^{(2)}, u_1^{(1)})$.

6. Show that the following function is primitive recursive:

$$f(0) = 0, \qquad f(1) = 1, \qquad f(2) = 2^{2^2}, \qquad f(3) = 3^{3^{3^3}}, \qquad \ldots \qquad f(n) = n^{n^{\cdot^{\cdot^{\cdot^{n}}}}} \ (n \text{ times})$$

SOLUTION Let $P(n,m) = n^{n^{\cdot^{\cdot^{\cdot^{n}}}}}$ (m times), then we define P recursively as follows

$$\begin{aligned} P(n,0) &= n = u_1^{(1)}(n) \\ P(n,m+1) &= (P(n,m))^n = Exp(P(n,m),n) \\ &= \Phi(Exp, u_3^{(3)}, u_1^{(3)})(n,m,P(n,m)) \end{aligned}$$

So, $P = R(u_1^{(1)}, \Phi(Exp, u_3^{(3)}, u_1^{(3)}))$. Now, notice that $f(n) = P(n,n)$, so $f = \Phi(P, u_1^{(1)}, u_1^{(1)})$.

7. Show that the following are recursive functions

   (a) $f(x,y) = \lfloor \log_x y \rfloor$
   SOLUTION Notice that $\log_x y = t \Rightarrow x^t = y$. Then $\lfloor \log_x y \rfloor = \min\{t \mid x^t \leq y \wedge x^{t+1} > y\}$. Also notice that the condition $x^t \leq y \wedge x^{t+1} > y$ can be rewritten as $x^t \dot{-} y = 0$ and $y \dot{-} x^{t+1} = 0$, or what is the same

   $$(x^t \dot{-} y) + (y \dot{-} x^{t+1}) = 0$$

   Then let $f'(t,x,y) = (x^t \dot{-} y) + (y \dot{-} x^{t+1})$, so

   $$f' = \Phi(\Sigma, \Phi(\circ d, \Phi(Exp, u_2^{(3)}, u_1^{(3)}), u_3^{(3)}), \Phi(\circ d, u_3^{(3)}, \Phi(Exp, u_2^{(3)}, \Phi(s, u_1^{(3)}))))$$

   And so, $f(x,y) = M[f'](x,y)$.

   (b) $g(x,y) = \lfloor \log_x y \rfloor + \lceil \sqrt[y]{x} \rceil$
   SOLUTION Notice that $\sqrt[y]{x} = t \Rightarrow t^y = x$. Then $\lceil \sqrt[y]{x} \rceil = \min\{t \mid (t-1)^y \leq x \wedge t^y > x\}$. Also notice that the condition $(t-1)^y \leq x \wedge t^y > x$ can be rewritten as

   $$((t-1)^y \dot{-} x) + (x \dot{-} t^y) = 0$$

   Then let $g'(t,x,y) = ((t-1)^y \dot{-} x) + (x \dot{-} t^y)$, so

   $$g' = \Phi(\Sigma, \Phi(\circ d, \Phi(Exp, \Phi(Pd, u_1^{(3)}), u_3^{(3)}), u_2^{(3)}), \Phi(\circ d, u_2^{(3)}, \Phi(Exp, u_1^{(3)}, u_3^{(3)})))$$

   So $g = \Phi(\Sigma, f, M[g'])$.

   (c) $g(x,y) = \lceil \sqrt[4]{x+y} \rceil$
   SOLUTION Notice that $\lceil \sqrt[4]{x+y} \rceil = \min\{t \mid (t-1)^4 \leq x+y \wedge t^4 > x+y\}$. Also notice that the condition $(t-1)^4 \leq x+y \wedge t^4 > x+y$ can be rewritten as

   $$((t-1)^4 \dot{-} (x+y)) + ((x+y) \dot{-} t^4) = 0$$

   Then let $h'(t,x,y) = ((t-1)^4 \dot{-} (x+y)) + ((x+y) \dot{-} t^4)$, so

   $$\begin{aligned} h' &= \Phi(\Sigma, \Phi(\circ d, \Phi(Exp, \Phi(Pd, u_1^{(3)}), four^{(3)}), \Phi(\Sigma, u_2^{(3)}, u_3^{(3)})), \\ &\quad \Phi(\circ d, \Phi(\Sigma, u_2^{(3)}, u_3^{(3)}), \Phi(Exp, u_1^{(3)}, four^{(3)}))) \end{aligned}$$

   So $h = M[h']$

8. Show that the following function is recursive primitive.

$$f(x,y) = \begin{cases} x+y & \text{if } x \text{ is an even number, multiple of 3.} \\ x \dot{-} y & \text{if } x \text{ is an odd number, multiple of 3.} \\ x & \text{otherwise.} \end{cases}$$

SOLUTION $f(x,y) = \circ D(\circ D(rem(x,3))).(\chi_{even}(x).(x+y) + \chi_{odd}.(x \dot{-} y)) + \circ D(rem(x,3)).x$, so

$$\begin{aligned} f &= \Phi(\Sigma, \Phi(\Pi, \Phi(\circ D, \Phi(\circ D, \Phi(rem, u_1^{(2)}, three^{(2)}))), \\ &\quad \Phi(\Sigma, \Phi(\Pi, \Phi(\chi_{even}, u_1^{(2)}), \Phi(\Sigma, u_1^{(2)}, u_2^{(2)})), \\ &\quad \Phi(\Pi, \Phi(\chi_{odd}, u_1^{(2)}), \Phi(\circ d, u_1^{(2)}, u_2^{(2)})))), \\ &\quad \Phi(\Pi, \Phi(\circ D, \Phi(rem, u_1^{(2)}, three^{(2)})), u_1^{(2)})) \end{aligned}$$

9. Consider the following function:

$$f(x,y) = \frac{x^2}{y}$$

Is it recursive primitive? In such case, write it as so, in other case, write it as recursive function (if possible). Justify in any case.

SOLUTION It is not recursive primitive because it is partial (it is not defined for $y = 0$). We cannot write it for the general case, however we can approximate it with $\lfloor \frac{x^2}{y} \rfloor$.

Notice that $\lfloor \frac{x^2}{y} \rfloor = t \Rightarrow x^2 \geq t.y \wedge (x+1)^2 < t.y$, or what is the same $(t.y \dot{-} x^2) + (x^2 \dot{-} t.y) = 0$.

Then let $g(t,x,y) = (t.y \dot{-} x^2) + (x^2 \dot{-} t.y)$, so

$$g = \Phi(\Sigma, \Phi(\circ d, \Phi(\Pi, u_1^{(3)}, u_3^{(3)}), \Phi(Exp, u_2^{(3)}, two^{(3)})), \Phi(\circ d, \Phi(Exp, \Phi(s, u_2^{(3)}), two^{(3)}), \Phi(\Pi, u_1^{(3)}, u_3^{(3)})))$$

Then $f = M[g]$.

Extra

10. Give the complexity (big $\mathcal{O}$ notation) of the Turing machines at exercise 2 and of the simulation at exercise 1.

SOLUTION

Exercise 1 Say the input of the machine to simulate is n and its complexity is N. There is a number of steps at the beginning to set up the machine, which depends *linearly* of the size of n (linearly since we have to copy the input (n) $k+1$ times, so we read and write $(k+1).n$ times). At each move we do k moves (one for head) and then $k.n$ to synchronize. So each move of the original machine takes $k + k.n$ moves of the new one. So, if the machine does in total N movements, we will do $(k + k.n).N$ plus the $(k+1).n$ for the setting. So, the number of steps is $k.(n+1).N + (k+1).n$. If we consider k fixed (i.e. a constant number), we can discard it and all the constants to take just the $\mathcal{O}$ notation, so we have $\mathcal{O}(n.N)$. That is, if the order of the original machine was n^2, the simulation will have an order n^3, and so on.

Exercise 2

Item (a) $\mathcal{O}(1)$. Constant. It just takes 3 steps.

Item (b) $\mathcal{O}(n)$. Lineal. It just passes through the tape once.

Item (c) $\mathcal{O}(\infty)$. It never terminates.

Item (d) $\mathcal{O}(n)$. Lineal. It do one pass and a half.

Item (e) $\mathcal{O}(1)$. Constant. It just takes 3 steps.

Item (f) $\mathcal{O}(n)$. Lineal. It runs a lineal machine and then a constant one.