Normal Algorithms

By further specifying the order of executions and some of the syntax of rewrite rules, one can obtain algorithms from string rewrite systems.

A *normal algorithm*, or Markov algorithm, is a string rewrite system with ordered rules: $R_1, \dots R_n$ over alphabet A where some of the rules may be regular rules $l \to r$ and some may be of the shape $l \to \cdot r$ in which case we say that the rule is *terminating*. For i between 1 and n, let $R_i = l_i \to r_i$ where r_i possibly starts with a "$\cdot$".

Given an input word w, one step of the algorithm corresponds to an application of the first applicable rule on the leftmost possible position. If no rule is applicable to w, the algorithm terminates with output w. The rule R_i is applicable to w iff $w = xl_iy$ for some words x and y. If a rule is applicable to w, let k be the smallest number such that the rule R_k is applicable to w. Moreover, let x be the shortest word such that $w = xl_ky$ for some word y. Then one step of the algorithm transforms w to $u = xr_ky$, which we denote by $w \xrightarrow{k} u$. The decoration on the arrow is for readability, to keep track of which rule was applied. If R_k is terminating, the algorithm terminates with result u. If R_k is not terminating, the algorithm proceeds with one step from u.

Here is an example of a normal algorithm (ε is the empty word):

$$
\begin{array}{rcl}
1.\ 00 & \to & \varepsilon \\
2.\ 01 & \to & \varepsilon \\
3.\ 10 & \to & \varepsilon \\
4.\ 11 & \to & \varepsilon \\
5.\ 0 & \to & \cdot 1 \\
6.\ 1 & \to & \cdot 1 \\
7.\ \varepsilon & \to & \cdot 0
\end{array}
$$

An example computation is: $11001 \xrightarrow{1} 111 \xrightarrow{4} 1 \xrightarrow{6} \cdot 1$
What does this algorithm compute / return ?

Normal algorithms are Turing complete. Intuitively this means: All problems that are algorithmically solvable, are solvable using a normal algorithm.
Note: You may need to extend the alphabet, also in the following tasks.

Please turn the page to see your tasks :-)

Task 1 Write a normal algorithm, that:

- Given a natural number, i.e., a word in alphabet 0,1,2,3,4,5,6,7,8,9, produces the smallest natural number formed with exactly the same digits (except for unnecessary 0's at the beginning). For example, given 109283, your algorithm should return 12389.
- Given a natural number, i.e., a word in alphabet 0,1,2,3,4,5,6,7,8,9, produces the largest natural number formed with exactly the same digits. For example, given 109283, your algorithm should return 983210.

$\triangleleft$

Task 2 Write a normal algorithm that given two natural numbers n and m in unary notation (n is represented by n $|$'s, for example 3 is represented by $|||$) delimited by a "$*$", returns the unary notation of the number $n+m$. For example, on input $||| * |||||$ your algorithm should return $||||||||$. $\triangleleft$

Task 3 Write a normal algorithm that given two natural numbers n and m in unary notation (n is represented by n $|$'s, for example 3 is represented by $|||$) delimited by a "$*$", returns the unary notation of the number $2n+m$. For example, on input $||| * |||||$ your algorithm should return $|||||||||||$. $\triangleleft$

Task 4* Write a normal algorithm that given a number in binary notation, produces the unary notation (as defined in Task 2) of the number. For example, on input 1011 the algorithm should terminate with output $|||||||||||$.

$\triangleleft$