

Week 8 Tutorial  
Lambda Calculus and Turing Machines

---

## 1 Warm Up

Reduce the following expressions to normal form:

1.  $(\lambda f. f (f 5)) (\lambda m. m + 1)$
2.  $(\lambda f x. f 5 x) (\lambda x y z. x + y * z) 2 3$

## 2 Church Encoding

In the lecture we defined true, false, and **if-then-else**. We show the normal way of defining **and** using **if**.

$$\begin{aligned} \text{true} &\stackrel{\text{def}}{=} \lambda x y. x \\ \text{false} &\stackrel{\text{def}}{=} \lambda x y. y \\ \text{if } e_1 \text{ then } e_2 \text{ else } e_3 &\stackrel{\text{def}}{=} e_1 e_2 e_3 \\ e_1 \text{ and } e_2 &\stackrel{\text{def}}{=} \text{if } e_1 \text{ then } e_2 \text{ else false} \end{aligned}$$

1. How should we define the ‘**or**’ operator?
2. Reduce the expression  $(\text{true and } (\text{false or true}))$  to normal form.

## 3 A Trivial Machine

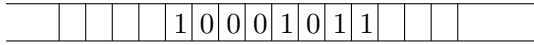
The following instructions specify a Turing Machine which recognizes the language given by the regular expression  $a^*b^*$ .

$\langle 0, \Lambda, \Lambda, S, Halt \rangle$   
 $\langle 0, a, a, R, 0 \rangle$   
 $\langle 0, b, b, R, 1 \rangle$   
 $\langle 1, b, b, R, 1 \rangle$   
 $\langle 1, \Lambda, \Lambda, S, Halt \rangle$

Draw the state change diagram for this machine and, assuming that the head is initially located at the leftmost end of the input, explain how it recognizes the strings  $abb$ , and  $bbb$ , but not the string  $ba$ .

## 4 Adding Two - in binary

The lecture notes gave a Turing machine that added one to a binary number. Construct a machine that adds two instead. Assume the data is represented on the tape with the least significant bit on the right. For example, the value 139 would be represented as:



Assume the head is initially somewhere over data.

## 5 Selective Removal

Give the state change diagram for a Turing machine that takes a tape containing a string of 'a's and 'b's and removes the 'b's. The initial string on the tape is terminated at each end by a blank symbol ( $\Lambda$ ). If the initial string has  $n$  a symbols then the final string will be exactly  $n$  'a's with no embedded blanks. (Hint: the original 'a's are all identical, they don't have to be in the same order in the final string).

## 6 Multiplication by 3

Specify a Turing machine which will multiply the binary number on its tape by 3. Make the same simplifying assumptions as in Question 4. When we do this operation manually we would work from right to left and compute at each bit position a new value and a carry, which can be 0, 1 or 2. The pair (new bit, carry-out) is a simple function of old bit value and carry-in. The following table captures this function.

	0	1	2
0	0,0	1,0	0,1
1	1,1	0,2	1,2
$\Lambda$	0,0	1,0	0,1

It is suggested that you try multiplying a few numbers by hand, using the above table, before starting on the Turing machine.