13.5 Turing Machines

A Turing machine consists of a finite state control unit and a tape divided into cells that expands infinitely in both directions. The control unit

- is in one state of finitely many different states at any one step
- has read and write capabilities on the tape as the control unit moves left and right on the tape

A Turing machine is the most general model of computation. They can model all computations that are performed on a computer.

Formal Definition of Turing Machine

A *Turing machine* $T = (S, I, f, s_0)$ consists of

- a finite set S of states
- an alphabet I containing the blank symbol B
- a partial function f ($f : S \times I \to S \times I \times \{R, L\}$)
  - The five-tuple (state, symbol, state, symbol, direction) corresponding to the partial function are called *transition rules*
- a starting state s_0

Transition in Turing Machines

If the control unit is in state s and if the partial function f is defined for the pair (s, x) with $f(s, x) = (s', x', d)$ (corresponding to the five-tuple (s, x, s', x', d)), the control unit will

1. Enter the state s'
2. Write the symbol x' in the current cell, thus erasing x
3. Move right one cell if $d = R$, or left one cell if $d = L$

If the partial function f is undefined for the pair (s, x), then the Turing machine T will *halt*.

At the initial state s_0, the control head is positioned either

- over the leftmost nonblank symbol on the tape
- over any cell if the tape is all blank

Recognizing Sets

A *final state* of a Turing machine T is a state that is not the first state in any five-tuple in the description of T using five-tuples.

Definition: Let V be a subset of an alphabet I. A Turing machine $T = (S, I, f, s_0)$ *recognizes* a string x in V^* if and only if T, starting in the initial position when x is written on the tape, halts in a final state. T is said to recognize a subset A of V^* if x is recognized by T if and only if $x \in A$.

Computing Functions

Turing machine as a computer of *number-theoretic functions* ($f : (n_1, n_2, \ldots, n_k) \to n_R$) where $n_1, \ldots, n_k$ and n_R are nonnegative integers.
To represent integers on a tape, we use *unary representations* of integers.

- Nonnegative integer n is represented by a string of $n + 1$ 1s. For example, 4 is represented by 11111.
- k-tuple $(n_1, n_2, \ldots, n_k)$ is represented by a string of $n_1 + 1$ 1s followed by an asterisk, followed by a string of $n_2 + 1$ 1s followed by an asterisk, and so on, ending with a string of $n_k + 1$ 1s. For example, (3,0,1,4) is represented by 1111*1*11*11111.

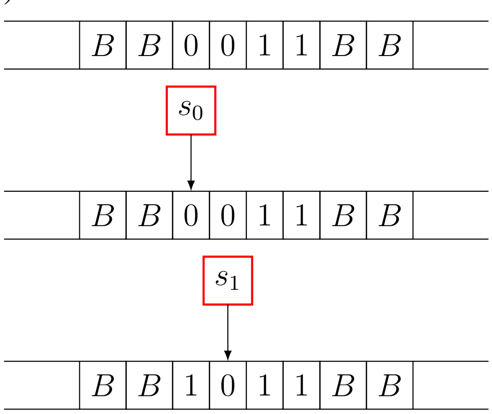
The Church-Turing Thesis

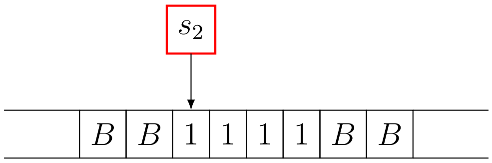
Given any problem that can be solved with an effective algorithm, there is a Turing machine that can solve this problem.

13.5 pg. 897 # 1

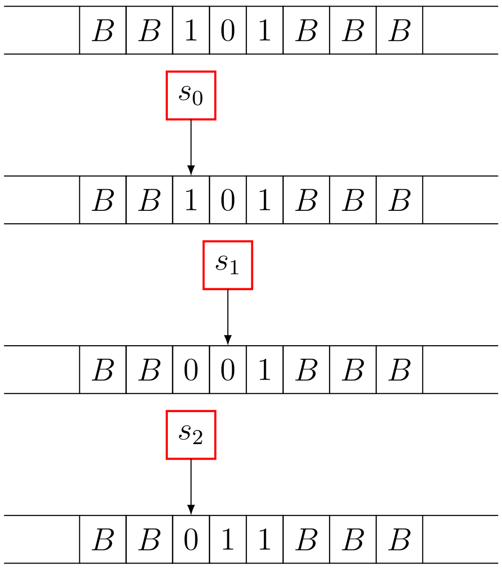
Let T be the Turing machine defined by the five-tuples: $(s_0, 0, s_1, 1, R)$, $(s_0, 1, s_1, 0, R)$, $(s_0, B, s_1, 0, R)$, $(s_1, 0, s_2, 1, L)$, $(s_1, 1, s_1, 0, R)$, and $(s_1, B, s_2, 0, L)$. For each of these initial tapes, determine the final tape when T halts, assuming that T begins in initial position.

a)

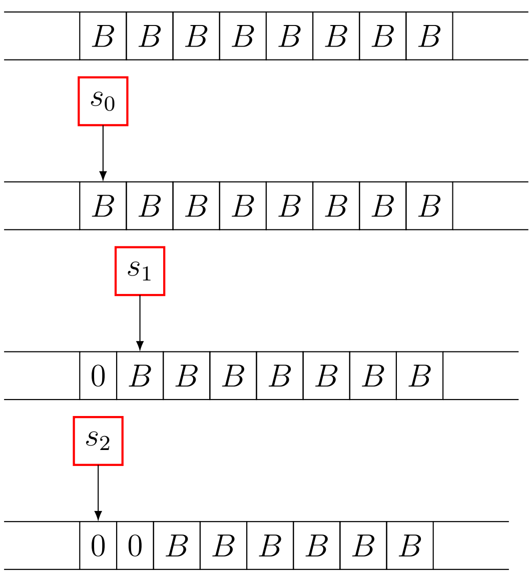




b)



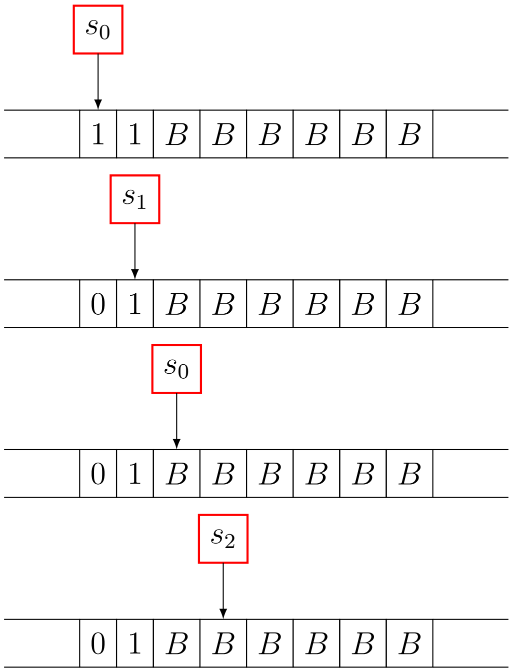
d)



13.5 pg. 898 # 3

What does the Turing machine described by the five-tuples $(s_0, 0, s_0, 0, R)$, $(s_0, 1, s_1, 0, R)$, (s_0, B, s_2, B, R), $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_0, 1, R)$, and (s_1, B, s_2, B, R) do when given

a) 11 as input?



b) an arbitrary bit string as input?

We first note that the tuple $(s_0, 0, s_0, 0, R)$ simply tells the machine to skip all beginning 0s when reading from the beginning. Only when the machine encounters a 1 is when the machine enters s_1 and writes a 0 to the current cell. Afterwards, we know from $(s_1, 0, s_1, 0, R)$ that in s_1, the machine will skip all 0s. Only when the machine encounters a 1 is when the machine enters s_0. Since the machine enters s_0 again, the process described before repeats. The machine will only halt when the the machine encounters a B. Meaning that this machine will read in a string and flip every other 1, starting with the first 1, in the bit string.

13.5 pg. 898 # 7

Construct a Turing machine with tape symbols 0, 1, and B that, when given a bit string as input, replaces the first 0 with a 1 and does not change any of the other symbols on the tape.

$(s_0, 1, s_0, 1, R)$ and $(s_0, 0, s_1, 1, R)$ will create the desired result. The first tuple allows the machine to keep scanning right until it encounters 0. If the machine encounters a 0, we enter s_1 and write a 1 in the cell. We do not need to define any functions for s_1 because we do not need to check the rest of the tape.

13.5 pg. 898 # 9

Construct a Turing machine with tape symbols 0, 1, and B that, when given a bit string as input, replaces all but the leftmost 1 on the tape with 0s and does not change any of the other symbols on the tape.

$(s_0, 0, s_0, 0, R)$, $(s_0, 1, s_1, 1, R)$, $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_1, 0, R)$. s_0 will represent our initial state where we are searching for the left most one as we start from the beginning of the symbols. Once we found a our leftmost 1, we can enter s_1 to change all the remaining 1s to 0s. This machine will halt when we encounter a B symbol.

13.5 pg. 898 # 11

Construct a Turing machine that recognizes the set of all bit strings that end with a 0.

$(s_0, 0, s_1, 0, R)$, $(s_0, 1, s_0, 1, R)$, $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_0, 1, R)$, and (s_1, B, s_2, B, R). s_0 represents that the last bit read was a 1. s_1 represents the last bit read was a 0. We can only accept when we encounter a B symbol while in s_1, so it can enter the final state, s_2.

13.5 pg. 898 # 13

Construct a Turing machine that recognizes the set of all bit strings that contain an even number of 1s.

$(s_0, 0, s_0, 0, R)$, $(s_0, 1, s_1, 1, R)$, $(s_1, 0, s_1, 0, R)$, $(s_1, 1, s_0, 1, R)$, and (s_0, B, s_2, B, R). s_0 represents the state where we have an even number of 1s. s_1 represents the state where we have an odd number of 1s. This machine will only accept when it encounters a B symbol while in s_0, so it can enter the final state, s_2.

13.5 pg. 898 # 19

Construct a Turing machine that computes the function $f(n) = n - 3$ if $n \geq 3$ and $f(n) = 0$ for $n = 0, 1, 2$ for all nonnegative integers n.

$(s_0, 1, s_1, B, R)$, $(s_1, 1, s_2, B, R)$, $(s_2, 1, s_3, B, R)$, $(s_3, 1, s_4, 1, R)$, $(s_1, B, s_4, 1, R)$, $(s_2, B, s_4, 1, R)$, and $(s_3, B, s_4, 1, R)$. The first 4 tuples apply for $n \geq 3$. These tuples will decrement the integer by 3 and then enter a final state s_4. The last 3 tuples are used when $n < 3$. We write a 1 when we encounter B because the other tuples have already erased the value in the previous cells and we want $f(n) = 0$. For example, if our input string is 11B, then the Turing machine will write $BB1$ to the tape, which evaluates to 0.