**CHAPTER 0-2**

**turing machines**

* In an effort to understand capabilities and limitations of machines, many researchers have proposed and studied various computational devices.
  + One of these is the Turing machine, which was proposed by Alan M. Turing in 1936 and is still used today as a tool for studying the power of algorithmic processes.
  + Turing’s goal was to provide a model by which the limits of “computational processes” could be studied.
* A Turing machine consists of a control unit that can read and write symbols on a tape by means of a read/write head.
  + The tape extends indefinitely at both ends and is divided into cells.

r/w head

Control unit

* + A tape cell can be empty (denoted by ), or it can contain any one of a finite set of symbols called the machine’s alphabet. Note that is never part of a machine's alphabet.
  + Initially, a finite segment of the tape contains the input string, and the rest of the cells are empty cells.
* At any time during a Turing machine’s computation, the machine must be in one of a finite number of states.
* A Turing machine’s computation begins in a special state called the start state and ceases when the machine reaches another special state known as the halt state.
* A Turing machine’s computation consists of a sequence of steps that are executed by the machine’s control unit.
  + - Each step consists of observing the symbol in the current tape cell (the one currently under the read-write head), writing a symbol in that cell, moving the r/w head one cell to the left or right or not at all, and possibly changing states.
    - The exact action to be performed during each step is determined by a program that tells the control unit what to do based on the machine’s current state and the contents of the current tape cell.

**An example TM**

* Let us consider an example Turing machine M. The alphabet for M consists of the symbols 0, 1, and \*. The tape of our machine might appear as follows:

\* 1 0 1 \*

r/w head

* + By interpreting a string of symbols on the tape as a binary number delimited by asterisks, we recognize that this particular tape contains the value 5.
* We will design M to increment such a value on the tape by 1.
* The states of M are START, ADD, CARRY, OVERFLOW, RETURN, and HALT.
* Here is the program of the control unit of M:

Current state current cell content value to write direction to move new state to enter

START \* \* left ADD

ADD 0 1 right RETURN

ADD 1 0 left CARRY

CARRY 0 1 right RETURN

CARRY 1 0 left CARRY

CARRY \* 1 left OVERFLOW

OVERFLOW \* right RETURN

RETURN 0 0 right RETURN

RETURN 1 1 right RETURN

RETURN \* \* no move HALT

* A TM's program can be better understood if we use a graphical notation:

(0/0/R)

(1/1/R)

RETURN

HALT

(0/1/R) (\*/\*/N)

(0/1/R) (/\*/R)

ADD

START

(\*/\*/L)

OVERFLOW

CARRY

(1/0/L)

(\*/1/L)

(1/0/L)

* We assume that M always begins in the START state with a legal number on the tape, and the r/w head is at an asterisk marking the right end of a string of 0s and 1s.
* As soon as it has started, M follows the program of its control unit and continues to execute until it reaches the HALT state.
* We can see that M, like any TM, can generate output (the contents on its tape when it halts).
* Note that a TM can halt in a non-halting state if no transition can be taken.
  + For example, if M is in the ADD state and the current symbol is \*, there is no move M can make. In such a case, the TM crashes and halt (there is a bug in the program).
  + In general, if the input is not well formed, the machine will crash.
* It is possible that a TM may never reach a halting state or crash, in which case it will loop forever.

**The Church-Turing thesis**

* The Turing machine M in the preceding example can be used to compute the successor function.
  + We need merely place the input value in its binary form on the machine’s tape, run the machine until it halts, and then read the output value from the tape.
  + A function that can be computed in this manner by a Turing machine is said to be Turing computable.
* Turing’s conjecture was that the Turing-computable functions were the same as the computable functions.
  + In other words, he conjectured that the computational power of Turing machines encompasses that of any algorithmic system or, equivalently, that the Turing machine concept provides a context in which solutions to all the computable functions can be expressed.
  + Today, this conjecture is often referred to as the Church–Turing thesis, in reference to the contributions made by both Alan Turing and Alonzo Church.
  + Since Turing’s initial work, much evidence has been collected to support this thesis, and today the Church–Turing thesis is widely accepted. That is, the computable functions and the Turing-computable functions are considered the same.

**HW**

1. Apply the TM M described in this handout, starting with the following initial status:

\* 1 1 0 \*

State=START r/w head

Show all the states of the TM leading up to the halt state.

1. Describe a TM that replaces a string of 0s and 1s with a single 0. You can assume the TM starts with a legal number on the tape. Note that there is no need for the machine to erase everything. You just need to make sure the machine halts with \*0\* immediately to the left of the r/w head. Give your answer using the graphical notation.
2. What function does the following TM compute?

Current state current cell content value to write direction to move new state to enter

START \* \* left SUBTRACT

SUBTRACT 0 1 left BORROW

SUBTRACT 1 0 right RETURN

BORROW 0 1 left BORROW

BORROW 1 0 right RETURN

BORROW \* \* right ZERO

ZERO 0 0 right ZERO

ZERO 1 0 right ZERO

ZERO \* \* no move HALT

RETURN 0 0 right RETURN

RETURN 1 1 right RETURN

RETURN \* \* no move HALT

You must also show the graphical notation of this TM.