

# An introduction to linear bounded machines

Jack Zhao Jin

Fall 2004, Denison University

## Part I: The introduction

Linear bounded machines are like Turing machines, they both have a set of states operating on a tape. Their only difference is that linear bounded machines have a tape with finite length, while Turing machines have a tape with infinite length. This paper will formally define what a linear bounded machine is and compare it to all the other machines we have studied over the semester.

## Part II: Formal definition

I will define a linear bounded machine as follows.  $M = (Q, \Sigma, \Gamma, \delta, q_0, S, E, F)$ , where  $M$  is a linear bounded machine. Inside the tuple we have

$Q$  as the finite set of states

$\Sigma$  as a finite set of alphabet

$\Gamma$  as finite a set of tape symbol

$\delta$  as a list of all the transition functions, such that  $\delta(q, X)$ , if defined, is a triple  $(p, Y, D)$ , where  $q$  is a state,  $X$  is a tape symbol,  $p$  is the next state in  $Q$ ,  $Y$  is the symbol in  $\Gamma$  that will replace the current symbol,  $D$  is the direction which the head moves, could be left, right, or stay where you are.

$q_0$  is the start state where the machine starts, a member of  $Q$

$S$  is the starting tape symbol to mark off the low end of the tape, and the machine will not be able to shift to the left beyond that

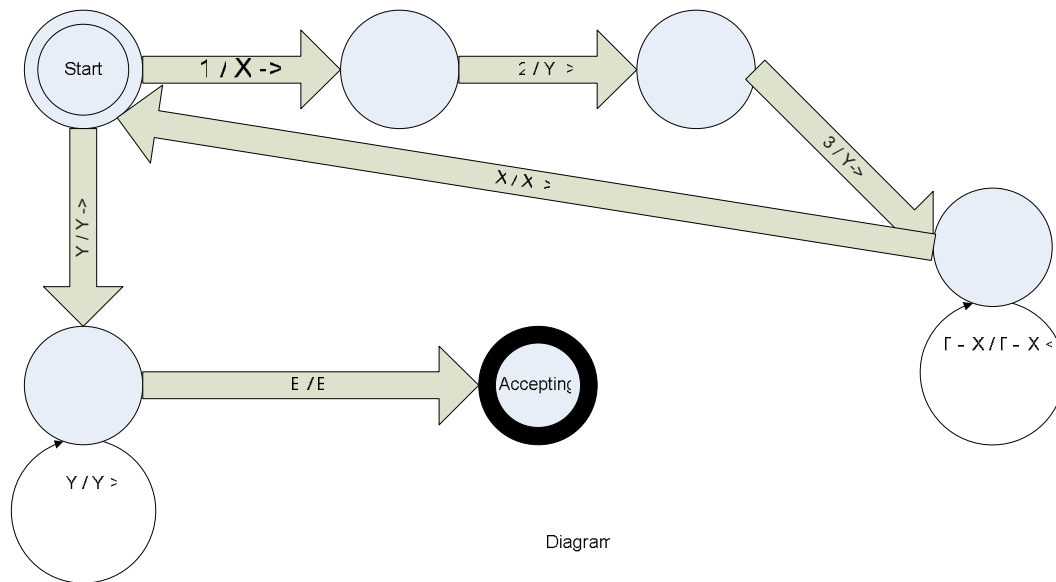
E is the ending tape symbol to mark off the higher end of the tape, and the machine will not be able to shift to the right beyond that

F is the accepting state in the machine, a member of Q

The tuple for this machine looks very similar to the tuple for a Turing machine. The only difference is that a Turing machine has a tape symbol B to represent the blank spaces on the tape, and a linear bounded machine has S and E to represent the start and the end of the tape. Note that the transition functions of the linear bounded machine are the same as the transition functions of a Turing machine.

### **Part III: Comparing linear bounded machines to push down machines and Turing machines**

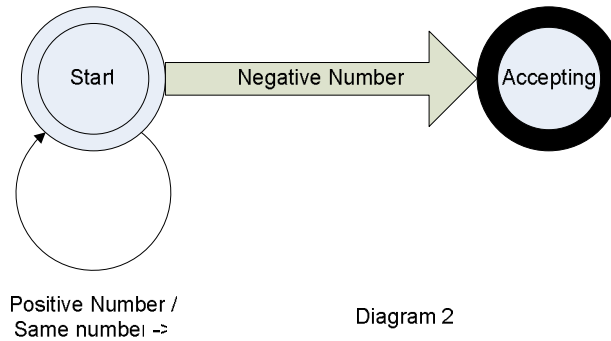
A push down machine can not accept a language such that  $L = \{1^i 2^i 3^i \mid i > 0\}$ . To show that a linear bounded machine is more powerful than the push down machine, I will demonstrate an example on how to use linear bounded machine to solve  $L = \{1^i 2^i 3^i \mid i > 0\}$ . Diagram 1 is an example of a linear bounded machine accepts language L. The machine places the input string into the tape with start and end symbol at the beginning and the end such that the tape looks like the following: S | 1 | ... | 1 | 2 | ... | 2 | 3 | ... | 3 | E. The start state of the machine begins reading the tape at the first position after the one. Please note that the language must reach the accepting state for the string to be accepted.



The difference between a Turing machine and a linear bounded machine is that the linear bounded machine has a finite tape length. It turns out a linear bounded machine can solve recursive problems, but not recursive enumerable problems. To show that a linear bounded machine can not solve a recursive enumerable problem, consider the following. Suppose machine  $M$  is a linear bounded machine capable of solving a particular recursive-enumerable problem. Since  $M$  is linear bounded, it implies that there are only a finite number of combinations between states, tape position, and tape. That implies the machine will go through all possible scenarios in a finite amount of time, and a conclusion to the problem will be drawn in a finite amount of time. This contradicts with the definition of recursive enumerable problems, that recursive enumerable problem does not end if it hasn't found the answer yet.

Here is another example, let's suppose we're using a linear bounded machine to solve the following problem: Find a natural number that's negative. Note that the problem is a recursive enumerable problem. A linear bounded machine can attempt to solve that problem with such a machine shown in diagram 2. If this is a diagram for a

Turing machine, it is the solution to solving such problem. Since this is a linear bounded problem, there is finite space on the tape, and therefore there can only be a finite number of executions before all the tape is used. Not all possible cases are examined under the linear bounded a model, so therefore a linear bounded machine fails to solve this particular problem.



## Part IV: Chomsky's hierarchy and context-sensitive grammars

In terms of Chomsky's hierarchy, linear bounded machines place between push down machines and Turing machines. As demonstrated above, a linear bounded machine could solve problems a push down machines can't solve, and a Turing machine can solve a series of problems that a linear bounded machine can't solve. It turns out all linear bounded machines can be represented using context-sensitive grammars. A context-sensitive grammar is similar to the context free grammar, except that it's possible to have statements such as  $AB \Rightarrow BC$ . A more formal way to describe context sensitive grammar is to say that  $X \Rightarrow Y$ , such that the number of variables of  $X$  is less or equal to the number of variables of  $Y$ .

## **Part V: Linear bounded machines and algorithms**

Linear bounded machines can be considered as an algorithm. Just like the Turing machine, the linear bounded machine has an accepting state. When you hit the accepting state, whatever is on the tape is the end result of an algorithm. Linear bounded machines can be used as sub-routines for other linear bounded machine or Turing machines as well.

Here is another example, suppose you have a linear bounded machine  $M_1$  traverse through the tape to the right until you hit the E symbol.  $M_1$  can be used as an algorithm by another machine  $M_2$  to go to the end of the table.  $M_1$  is an algorithm can be used by  $M_2$ .

## **Part VI: Determinism in linear bounded machines**

Non-deterministic linear bounded machine does not exist. For a non-deterministic linear machine to be non-deterministic, it must be able to simulate multiple tapes, such that each tape simulates a particular case. This action can not be performed on a linear bounded machine since the tape size is pre-determined, and number of tapes is pre-determined. Considering a non-deterministic state maybe executed any number of times, the number of tapes to allocate before hand can not be determined. The only way to simulate non-determinism is with a tape that has infinite length.

## **Part VII: Conclusion**

Linear bounded machines are not as powerful as a Turing machine, but almost as powerful. Linear bounded machines are very similar to today's real machines, since no real machine truly has "infinite tape". The goal of this project is to design a machine

that's not covered in class, and this paper is written based on my own understanding of linear bounded machines.