CS21004 Formal Languages and Automata Theory, Spring 2019–2020

Nondeterministic Turing Machines

1. Determine the (nondeterministic) running time of the compositeness test.

Solution The copying phase takes $\Theta(d)$ time. Subsequently, the division stage takes $\Theta(n)$ time, because this stage involves about n/d subtractions each taking $\Theta(d)$ time. Since $2 \leqslant d \leqslant n-1$ (so $d = \mathrm{O}(n)$), the overall running time is $\Theta(n)$.

2. Why cannot the nondeterministic compositeness test be used to accept the language $\{a^p \mid p \text{ is a prime}\}$?

Solution The acceptance criterion of an NTM is the existence of a sequence of guesses leading to the accept state. The primality of p requires that p is not divisible by all d in the range $2 \leqslant d \leqslant p-1$. The existence of one non-divisor d or p cannot immediately lead to a conclusion. All d in the range mentioned above must be non-divisors. So the computation has to be sequential for all d.

There exist nondeterministic primality tests more sophisticated than the simple search for divisors. A discussion on this topic is well beyond the scope of this course. You are likely to learn something called *Pratt certificates* in a course on Computational Complexity.

3. Design an NTM to accept the language

$$\{wxyxz \mid w,x,y,z \in \{0,1\}^*,\ |x| = 100\}.$$

Solution Let us design a two-tape NTM for accepting the given language. The working of the machine has the following stages.

1. Skip Stage 1: Starting from the beginning of the input, keep on moving right, and at some point nondeterministically jump to the next stage. If the input ends during the rightward march, reject.
2. Copy Stage: Copy the next 100 symbols from the input to the second tape. If the input ends before the copying all the symbols, reject.
3. Skip Stage 2: Starting from the position at the end of the copy stage, keep on moving right, and at some point nondeterministically jump to the next stage. If the input ends during the rightward march, reject.
4. Compare Stage: Compare the next 100 symbols from the input with the content of the second tape. If all symbols match, accept. If there is a mismatch, or the input ends before the completion of 100 comparisons, reject.

4. Suppose that the simulator D uses its first tape as a stack of configurations of N. Will the simulation work? Justify.

Solution No. Using a stack means performing a DFS traversal in the computation tree. There may be accepting paths (must be finite) starting from the initial configuration and ending in a configuration with the accept state. Alongside that, there may exist looping behavior for some sequence of nondeterministic choices. If the DFS exploration is stuck on such an infinite computation sequence, it can never track back and locate the finite accepting path.

5. A TM M has a two-way infinite tape. Initially, all cells on the tape are blank. Only one cell is storing the symbol #. The head of M is pointing to a blank. The task of M is to locate the cell storing #. Propose a strategy for doing this

(a) if M is a DTM,

Solution M locates the # by making a to-and-fro movement. First, it moves one step left, then two steps right, then three steps left, then four steps right, and so on. Since M has only one tape, it cannot maintain a counter on a separate tape. What instead it can do is it can mark the cells as visited. It starts by marking its initial position as visited. Then during each leftward or rightward run, it locates the first unvisited cell, marks it visited, and changes the direction of exploration.

(b) if M is an NTM.

Solution M nondeterministically guesses the direction to which # resides from the initial position. M then keeps on moving in that direction until it reaches the cell containing the #.

(c) Compare the deterministic and nondeterministic running times.

Solution Let d be the distance of the cell containing the # from the initial position of the head.

If the # is to the left of the initial position, the number of steps for the DTM will be $1+2+3+4+\cdots+(2d-2)+(2d-1)=d(2d-1)$. If the # is to the right of the initial position, the number of steps will be $1+2+3+4+\cdots+(2d-2)+(2d-1)+2d=d(2d+1)$. In both the cases, the running time is $\Theta(d^2)$.

For the NTM, the number of steps is d for the correct guess. The incorrect guess leads the head to move in the wrong direction indefinitely. But we can take $\Theta(d)$ as the nondeterministic running time of the NTM.