CS 6810: Theory of Computing **Fall 2023**

Lecture 2: Aug 24, 2023

Lecturer: Eshan Chattopadhyay *Scribe: Ellie Fassman*

1 Universal Turing Machines

1.1 Notation

A turing machine (TM) **M** can be described by a bitstring, that is, $\lfloor \mathbf{M} \rfloor$ denotes the description of **M**. $\lfloor \mathbf{M} \rfloor \in \{0,1\}^*$ where the description contains Q: the state space, Σ: the input alphabet, Γ: the tape alphabet, k: the number of work tapes, and δ: the transition function. To further clarify δ,

$$\delta : Q \times \Gamma^{k+2} \to Q \times \Gamma^{k+1} \times \{L, R, S\}^{k+2}$$

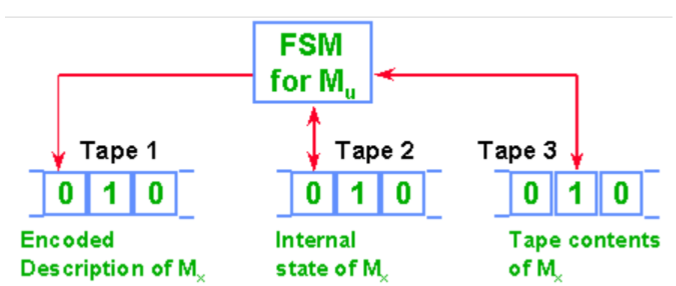
where δ specifies how **M** will behave upon being in state Q and reading the $k+2$ elements of Γ (one element on each of its tapes). δ will specify the state Q that **M** enters, the $k+1$ elements of Γ that **M** will write on its tapes (every tape except the input tape, which we restrict to be read-only), and the direction on each tape that **M** will move (this is the $k+2$ specifications of $\{L, R, S\}$). Note that $\lfloor \mathbf{M} \rfloor \in \{0,1\}^*$ is constant, that is, the description of **M** does not depend on the input length.

1.2 What is a Universal TM?

A universal TM **U** takes in input $\lfloor \mathbf{M} \rfloor$, x and simulates **M** on x. The states of **U** cannot depend on its input (specifically the states of **U** cannot depend on $\lfloor \mathbf{M} \rfloor$). The amazing thing about universal TM's is that we can specify a **U** whose description is fixed but can simulate any TM **M** on input x. That is, if **M** outputs 1 on x, **U** outputs 1 on $\lfloor \mathbf{M} \rfloor$, x. Likewise, **U** will output 0 when **M** outputs 0 on x. For simplicity we will only be considering TM's that halt on input x.

Theorem 1.1. *(Informal)* ***U*** *simulates* ***M*** *on x with a logarithmic overhead. That is, if* ***M*** *halts on x after $T(|x|)$ steps, then* ***U*** *halts on $\lfloor \boldsymbol{M} \rfloor$, x after CTlog(T) steps.*

A proof of this can be found at the end of chapter 1 in the Arora-Barak textbook. Below is a diagram hinting at how a TM $\mathbf{M}_u$ simulates the whole computation of $\mathbf{M}_x$ on its work tapes.



2 NP (Nondeterministic Polynomial Time)

Definition (NP Relation): R $\subseteq \{0,1\}^* \times \{0,1\}^*$ is an NP relation if there exists a verifier TM **V** that runs in polynomial time such that:

- If (x,y) $\in$ R then $|y| \leq C|x|^C$
- **V**(x,y) = 1 if (x,y) $\in$ R and **V**(x,y) = 0 otherwise.

There are two versions of NP. The **decision version of NP** asks: given x, does there exist a y such that (x,y) $\in$ R? The **search version of NP** asks: given x, find y (if any) such that (x,y) $\in$ R. We will see that if you can solve the decision version of this question for a given R, you can also solve the search version.

Claim 2.1. *Given any problem P_1 in search-NP, there is a problem P_2 in decision-NP such that: If there exists a TM **M** that solves P_2 in T(n) time, then there exists a TM **M*** that solves P_1 in $poly(n) \cdot T(poly(n))$ time.*

Suppose P = (decision-)NP. The claim above then implies that all problems in search-NP can be solved in polynomial time.

Proof. We are given NP relation R_1 where P_1 is: given x, find y (if any) such that (x,y) $\in R_1$. Define R_2 as: ((x,w),z) $\in R_2$ if (x,w $\circ$ z) $\in R_1$. P_2 is the decision problem for R_2 and we want to build an algorithm to solve P_1 (search) given an algorithm, say $\mathcal{A}_2$, which solves P_2 (decision).

To clarify the goal, given x and we want to find the certificate y (if it exists). We must build y up bit by bit. This is because in R_2 we are given (x,w) and we are asking if there exists a z for which $(x, w \circ z) \in R_1$ but we need to find this z for R_1. This can be accomplished by running algorithm $\mathcal{A}_2$ on (x, ϵ). If the answer is No, then we're done (no y exists for which $(x, y) \in R_1$). If the answer is yes, we know that y exists and thus either begins with a 0 or a 1. Niw, iteratively we can run $\mathcal{A}_2$ on (x, 0) - if the answer is Yes, the we next run $\mathcal{A}_2$ $(x, 00)$ and so on. If the answer is No, we run $\mathcal{A}_2$ on $(x, 10)$.

The bound on the running time is straightforward from the fact that $|y| \leq c|x|^c$, for some constant c. $\square$

3 Nondeterministic Turing Machines

WA non-deterministic TM **N** is the same as a multi-tape TM but has two transition functions, δ_0 and δ_1.

$$\delta_0, \delta_1 : Q \times \Gamma^{k+2} \to Q \times \Gamma^{k+1} \times \{L, R, S\}^{k+2}$$

N accepts input x if there exists any sequence of transition functions such that **N** halts on x and writes 1 on its output tape. The time complexity of **N** is T(n) if every sequence of δ_0 and δ_1 leads **N** to halt on x in T($|x|$) steps.