BU CS 332 – Theory of Computation

Lecture 12:

- TM Variants
- The Church-Turing thesis
- Universal DFAs, CFGs, TMs

Reading:

Sipser Ch. 3.2, 4

Ran Canetti

October 20, 2020

How Robust is the TM Model?

Does changing the model result in different languages being recognizable / decidable?

Short answer: No....

Longer answer:

Last week: TMs are equivalent to…

- TMs with “stay put”
- TMs with 2-way infinite tapes
- Multi-tape TMs

Last week: TMs are equivalent to…

- TMs with “stay put”
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Non-deterministic TMs

Non-deterministic TMs

At any point in computation, may non-deterministically branch. Accepts iff there exists an accepting branch.

Transition function $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R, S\})$

Ex. NTM for $\{w \mid w$ is a binary number representing the product of two positive integers $a, b\}$

a b

$a \cdot b \stackrel{?}{=} w$

Non-deterministic TMs

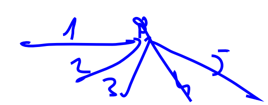
Theorem: Every nondeterministic TM N has an equivalent deterministic TM M

$L(M) = L(N)$

Proof idea:

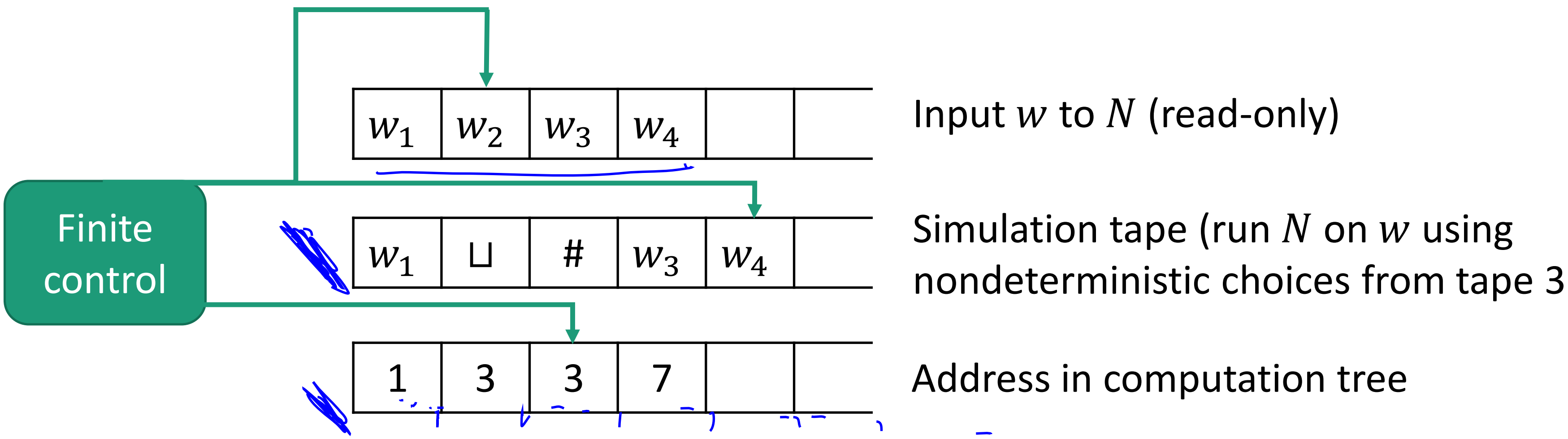
- Order the different choices at each step of N's computation
- Represent a specific computational path of N via a sequence of numbers, representing the choices:

2, 3, 4, 1, 28, … …

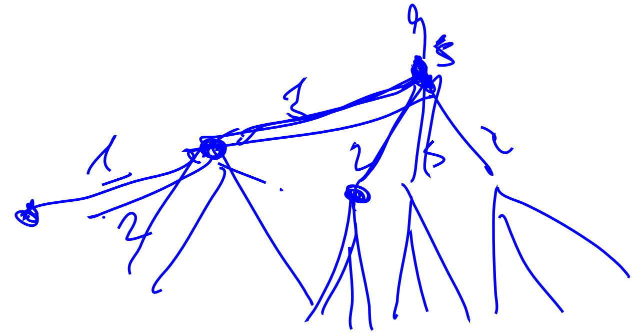


Non-deterministic TMs

- Simulate N using a 3-tape TM M:



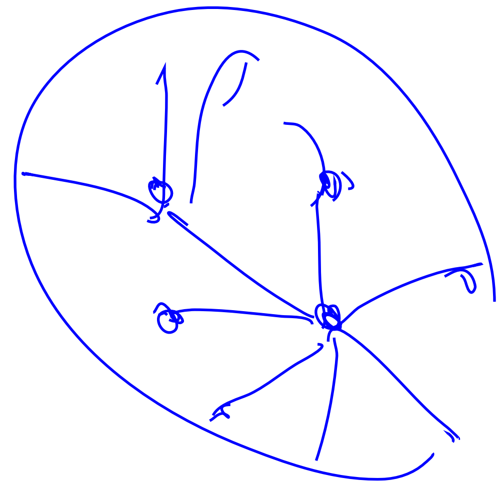
$q_0 \; q_1 - q_{12}, \; q_a \; q_r$



TMs are equivalent to…

- TMs with "stay put"
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Nondeterministic TMs
- Random access TMs
- 3D TMs
- Cellular automata
- Coin-tossing TMs
- Quantum TMs

…



The equivalence of these models is a mathematical theorem

Church-Turing *Thesis v. I*: Each of these models captures our intuitive notion of algorithms

The equivalence of these models is a mathematical theorem

Church-Turing *Thesis v. I*: Each of these models captures our intuitive notion of algorithms

Church-Turing *Thesis v. II*: Any physical computation process can be simulated on a TM.

The equivalence of these models is a mathematical theorem

Church-Turing *Thesis v. I*: Each of these models captures our intuitive notion of algorithms

Church-Turing *Thesis v. II*: Any physical computation process can be simulated on a TM.

The Church-Turing Thesis is not a mathematical statement!

Universal computation

- Can we encode algorithms as data?
- Can we generically run a given algorithm on a given input?

Universal computation

- Can we encode algorithms as data?
- Can we generically run a given algorithm on a given input?
- An algorithm that does that is a universal algorithm.

$A(\langle B \rangle, x) = B(x)$

Universal FDAs

DFA

Design a TM which takes as input a DFA D and a string w, and determines whether D accepts w

How should the input to this TM be represented?

Let $D = (Q, \Sigma, \delta, q_0, F)$. List each component of the tuple separated by ;

- Represent Q by ,-separated binary strings
- Represent Σ by ,-separated binary strings
- Represent $\delta : Q \times \Sigma \ \rightarrow Q$ by a ,-separated list of triples (p, a, q), …

Denote the encoding of D, w by $\langle D, w \rangle$

Representation independence

Existence of a universal algorithm (TM) is not affected by the choice of encoding.

Representation independence

Existence of a universal algorithm (TM) is not affected by the choice of encoding.

Why? A TM can always convert between different encodings

For now, we can take $\langle \quad \rangle$ to mean "any reasonable encoding"

A "universal" algorithm for recognizing regular languages

$A_{\mathrm{DFA}} = \{\langle D, w\rangle \mid \mathrm{DFA}\ D \text{ accepts } w\}$

Theorem: A_{DFA} is decidable

Proof sketch: Define a TM M which on input $\langle D, w\rangle$:

1. Check if $\langle D, w\rangle$ is a valid encoding (reject if not)
2. Simulate D on w, i.e.,
   - Tape 2: Maintain w and head location of D
   - Tape 3: Maintain state of D, update according to δ
3. Accept iff D ends in an accept state

Are the following languages decidable?

$A_{\text{DFA}} = \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$ ✓ Decidable

$A_{\text{NFA}} = \{\langle N, w\rangle \mid \text{NFA } N \text{ accepts } w\}$? Decidable

NFA $\Rightarrow$ DFA

N D

$L(N) = L(D)$

Recursive Languages

R

Are the following languages decidable?

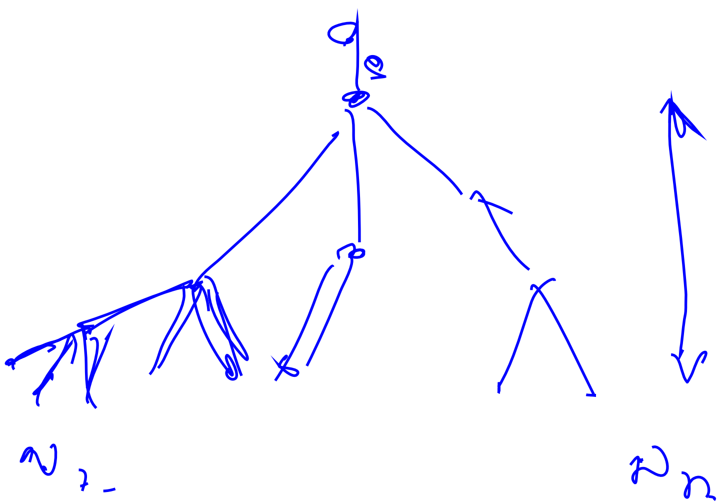
$A_{\mathrm{DFA}} = \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$ D

$A_{\mathrm{NFA}} = \{\langle N, w\rangle \mid \text{NFA } N \text{ accepts } w\}$ D

$A_{\mathrm{CFG}} = \{\langle G, w\rangle \mid \text{CFG } G \text{ generates } w\}$

$q_0 \to q_{acc}$

$q_1 \to q_0$



Univrersal CFG Generation

Theorem: $A_{\text{CFG}} = \{\langle G, w\rangle \mid \text{CFG } G \text{ generates } w\}$ is Turing-recognizable

Proof idea: Define a TM M recognizing A_{CFG}

On input $\langle G, w\rangle$

1. Enumerate all strings that can be generated from G
   (i.e., all length-1 derivations, all length-2 derivations, …)

2. If any of these strings equal w, accept

Fact: For any CFG with k states and any word $w \in L(CFG)$ with $|w| = n$, there exists a derivation tree for w with depth $\leq k \cdot n$

$\Rightarrow A_{CFG}$ is in R

Reminder: R = decidable languages

RE = recognizable languages

Recursively enumerable

Universal CFG Generation

Theorem: $A_{\mathrm{CFG}} = \{\langle G, w\rangle \mid \text{CFG } G \text{ generates } w\}$ is decidable

Encoding TMs

$M = (Q, \Sigma, \Gamma, q_s, q_a, q_r, \delta)$

$Q = \{0, \ldots, \}$

q_s

$\Sigma = \{0 \ldots$

$\Gamma = \{0 \ldots 1 \ldots\}$

$L = 0 \quad R = 1$

δ:

(curr state, read, new state, new symbol, direction)

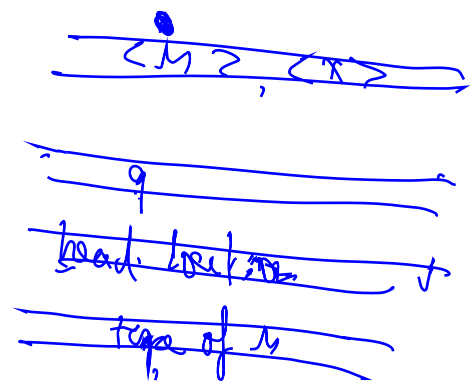
$\delta_1 = \delta_{11} \ldots \delta_{25}$

What about Universal TMs?

$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input } w\}$

is Turing-recognizable

The following "universal TM" U recognizes A_{TM}



On input $\langle M, w\rangle$:

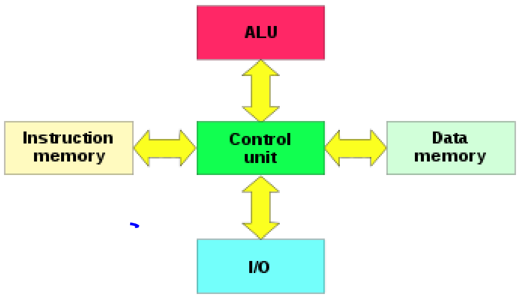
1. Simulate running M on input w
2. If M accepts, accept. If M rejects, reject.

More on the Universal TM

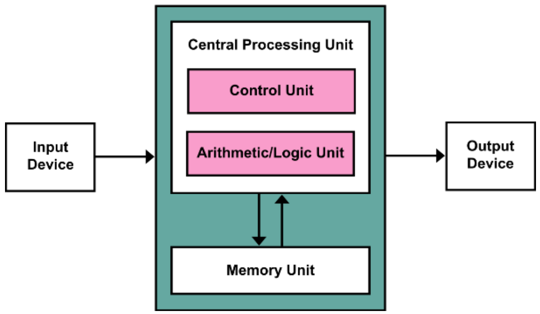
"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine **U** is supplied with a tape on the beginning of which is written the S.D ["standard description"] of some computing machine **M**, then **U** will compute the same sequence as **M**."

- Turing, "On Computable Numbers…" 1936

- Foreshadowed general-purpose programmable computers
- No need for specialized hardware: Virtual machines as software



Harvard architecture:
Separate instruction and data pathways



von Neumann architecture:
Programs can be treated as data