COMPSCI 501: Formal Language Theory

Lecture 12: Variants of Turing Machines

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

19 February 2019

Recap: Recognize vs. Decide

For given initial tape contents, a Turing machine could

- halt in the accept state
- halt in the reject state
- **loop** forever (not halt)

Recognize: accept all and only the strings in the language
(reject or loop otherwise)

Decide: TM never loops (either accepts or rejects)

Deciding is stronger than recognizing.

Some Turing-recognizable languages are not Turing-decidable
(will see later).

From DFAs to Turing Machines

Extra capabilities of TM:

1. can move both ways on tape (revisit input)
2. can **write** on tape
3. can use **additional unlimited memory**

Adding just (1): two-way automata (recall CS 250)
Result (surprising?): same as normal DFA.

Add (1) and (2): **linear bounded automata**
context-sensitive languages

Do Details Matter?

A Turing Machine is a 7-tuple ...

What can change in the definition while keeping the essence?

Automata: DFA / NFA / ε-transitions: same

Pushdown automata:
Normalizing moves (either push or pop): same expressiveness
Nondeterminism mattered !

"Robustness" of definition

First Change: Stay Put

Our looping constructions so far often "overshoot" by one:

"find first symbol of certain kind"
when found, must do something: move left or right
but perhaps we want to stay / start a sweep there

Change transition function to $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$
S = stay in place

Does this change anything?

Clearly we can model "stay put" by inserting move right, then left.
why not the other way around?

Multitape Turing Machines

- Each tape has its own read/write head
- Input is initially on tape 1, other tapes blank
- Heads move/read/write simultaneously

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

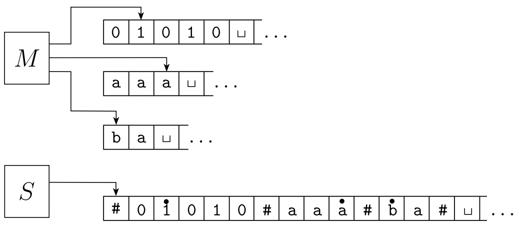
One transition:

$\delta(q_i, a_1, \dots a_k) = (q_j, b_1, \dots, b_k, R, L, \dots, R)$

Does this add expressive power?

Proving Equivalence: Simulation

Show that multitape TM M can be simulated by single-tape TM S



- Concatenate all tape contents onto one, with markers between
- Mark head locations with new "dotted" alphabet symbols
- Head moves right, simulating a round of moves
- If any tape needs extending, shift remaining contents right

Nondeterministic Turing Machines

New transition function allows choice:
$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R\})$

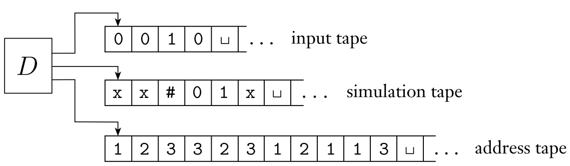
$\Rightarrow$ computation *tree*, accept if some branch accepts

Theorem: Every nondeterministic TM has an equivalent deterministic TM

Will prove by constructing a *three-tape* TM that simulates all nondeterministic choices of N

Simulating a Nondeterministic TM

Computation tree potentially infinite: $\Rightarrow$ traverse breadth-first ensures shortest accepting computation will be found

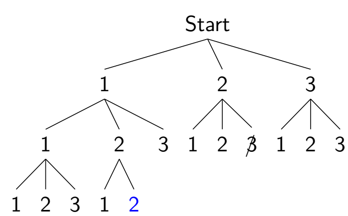


Keep:

1. Unmodified copy of input
2. Working tape (simulation)
3. Place in computation tree (initially: root)

Figure: Sipser, 3rd ed.

Encoding Computation Branches



Assume max. degree (nondeterminism) = 3. Current path: 1 2 2.
Next paths: 1 2 3, 1 3 1, 1 3 2, etc.

- Copy original input (tape 1) to tape 2
- Simulate run of N on computation string from tape 3
- Update tape 3 to next string

Enumerators

Turing-recognizable languages are also called **recursively enumerable**

Think Turing machine with attached printer
prints all strings in the language
may print infinite list, or halt if language is finite
repetitions are allowed (don't matter)

Theorem: A language is Turing-recognizable iff some enumerator enumerates it.

RE - Turing Equivalence

"$\Rightarrow$": Obtain TM from enumerator E

- Run E. Compare each generated string with input w
- If w appears in the output of E, accept.

"$\Leftarrow$": Obtain enumerator E from TM

Need to run TM on all strings of Σ^*: $s_1, s_2, s_3, \ldots$
Can't go "depth-first" $\Rightarrow$ run **diagonally**:
for 1 step on s_1
for 2 steps on s_1, s_2
for 3 steps on s_1, s_2, s_3, etc.
print any string if accepted (repetition OK)

Simulates running M in parallel on all input strings