Lecture 13: Turing Machines

The Story So Far

- Regular Languages
  - Described by DFA, NFA, RegExp, RegGram
- Context-Free Languages
  - Described by CFG, NDPDA

Violates Pumping Lemma for RLs
Violates Pumping Lemma for CFLs

The Story So Far (Simplified)

- Regular Languages
- Context-Free Languages

Computability Story: This Week

- Languages recognizable by any mechanical computing machine

Computability Story: Next Week

- Undecidable Problems
- Decidable Problems
- Recognizable Languages

Computability → Complexity (April)

- Decidable Problems
- Recognizable Languages
- NP
- P

Problems that can be solved by a computer (eventually).
Problems that can be solved by a computer in a reasonable time.

Note: not known if $P \subseteq NP$ or $P = NP$
Exam 1

Problem 4c: Prove that the language \( \{0^n1^{n^2}\} \) is not context-free.

Lengths of strings in \( L \):

\[
\begin{align*}
  n = 0 & \quad \Rightarrow 0^01^0 = 0 \\
  n = 1 & \quad \Rightarrow 1^11^1 = 2 \\
  n = 2 & \quad \Rightarrow 2^21^4 = 6 \\
  n = 3 & \quad \Rightarrow 3^31^9 = 12 \\
  \vdots \\
  n = k & \quad \Rightarrow k^k1^{k^2} 
\end{align*}
\]

Pumping lemma for CFLs says there must be some way of picking \( s = uvxyz \) such that \( m = |v| + |y| > 0 \) and \( uv^ixy^iz \in L \) for all \( i \).

So, increasing \( i \) by 1 adds \( m \) symbols to the string, which must produce a string of a length that is not the length of a string in \( L \).

Recognizing \( \{0^n1^{n^2}\} \)

DPDA with **two** stacks?

DPDA with **three** stacks?

...?

Can it be done with 2 Stacks?

3-Stack DPDA Recognizing \( \{0^n1^{n^2}\} \)

Simulating 3-DPDA with 2-DPDA
Simulating 3-DPDA with 2-DPDA$^+$

- $\text{pop green}$
- $\text{push}(S)$
- $\text{push}(\text{pop}_A(\#))$
- $\text{push}(\text{pop}_B(\#))$
- $\text{push}(\text{pop}_B(\#))$
- $\text{res} = \text{pop}_A()$
- $\text{push}(\text{pop}_B(\#))$
- $\text{push}(\text{pop}_B(\#))$
- $\text{pop}_B(\#)$

2-DPDA + Forced $\varepsilon$-Transitions

- Need to do lots of stack manipulation to simulate 3-DPDA on one transition: need transitions with no input symbol (but not nondeterminism!)

Impact of Forced $\varepsilon$-Transitions

- What is the impact of adding non-input consuming transitions?
- DPDA in length $n$ input: runs for $\leq n$ steps
- DPDA+$\varepsilon$ in length $n$ input: can run forever!

Is there any computing machine we can’t simulate with a 2-DPDA$^+$?

What about an NDPDA?

- Use one stack to simulate the NDPDA’s stack.
- Use the other stack to keep track of nondeterminism points: copy of stack and decisions left to make.

Turing Machine?
Turing Machine

Infinite tape: $\Gamma^*$
Tape head: read current square on tape, write into current square, move one square left or right

FSM: like PDA, except:
- transitions also include direction (left/right)
- final accepting and rejecting states

Turing Machine Computing Model

Initial configuration:

```
x x x x x x x _ _ _ _ ...
```

FSM

TM Configuration: $\Gamma^* \times Q \times \Gamma^*$
- tape contents left of head
- current FSM state
- tape contents head and right

TM Computing Model

$$\delta^*: \Gamma^* \times Q \times \Gamma^* \to \Gamma^* \times Q \times \Gamma^*$$

$u, v \in \Gamma^*, a, b \in \Gamma$

- $\delta^*(ua, q, q) = (uac, q_r, c)$
- $\delta^*(ua, q, R) = (u, q, acv)$

Also: need a rule to cover what happens at left edge of tape

Thursday’s Class

- Robustness of TM model
- Church-Turing Thesis

Read Chapter 3:
- It contains the most bogus sentence in the whole book. Identify it for +25 bonus points (only 1 guess allowed!)