COMS W3261 — Lecture 8

CFPL & Turing Machines.

Teaser: Which actor played computer scientist Alan Turing in the 2014 film *The Imitation Game*?

Bonus: Who played cryptographer/codebreaker Joan Clarke?

Benedict Cumberbatch

Kiera Knightly

Announcements: HW #4 due today, 7/26/21 @ 11:59 PM EST
HW #5 out now, due Monday, 8/2 @ —"—
HW #3 graded soon.
Final exam post up on Ed.

Readings: (2.3 CFPL)
3.1 Turing Machines,
Turing-Recognizability & Decidability

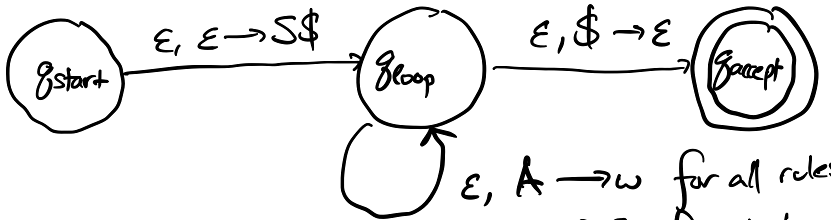
Today:
1. Review (PDA $\Longleftrightarrow$ CFG)
2. CFPL Examples & Proof
3. TMs
4. Decidability & Recognizability (for TMs).

(Problem Discussion?)

1. Review.

CFG $\Longleftrightarrow$ PDA.

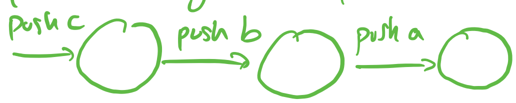
We sketched out how to turn any CFG into a PDA:
(For some grammar $G = (V, \Sigma, R, S)$):



$\varepsilon, A \to w$ for all rules $A \to w$ in R

$a, a \to \varepsilon$ for all terminals $a \in \Sigma$

we pushed strings — to push abc

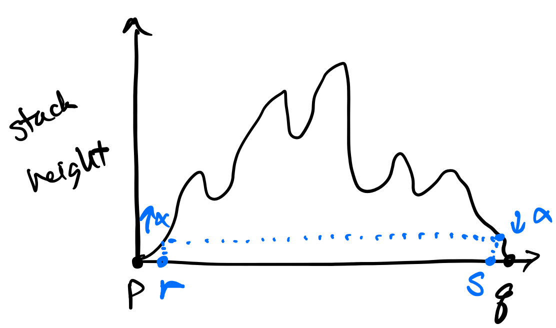


We also sketched how to turn any PDA into a CFG:

A_{pq} : variable that generated all strings corresponding to a computation from p to q in our PDA, with empty stacks before and after.

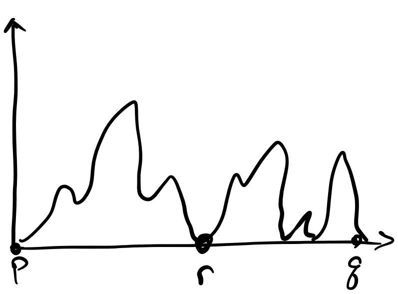
We argued then that $A_{q_0 q_{accept}}$ would derive all strings.

<u>Case 1:</u>



$A_{pq} \to a A_{rs} b$, where a takes us $p \to r$, and b takes us $s \to q$.

<u>Case 2:</u>



$A_{pq} \to A_{pr} A_{rq}$.

Punchline: A language is CF if and only if it is recognized by some PDA.

Theorem (PL for context-free languages.) If L is context-free, then there exists some "pumping length" p such that, for all $s \in L$, $|s| \geq p$, s can be divided into five strings $s = uvxyz$ such that

(1) $uv^ixy^iz \in L$ for all $i \geq 0$.

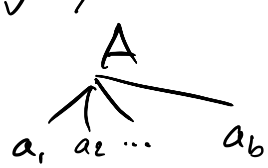
(2) $|vy| > 0$,

(3) $|vxy| \leq p$.

Proof idea: if we repeat a variable in some derivation, we can "loop" by repeating the sequence of rules from the variable to itself many times.

Proof. (PL for CFLs).

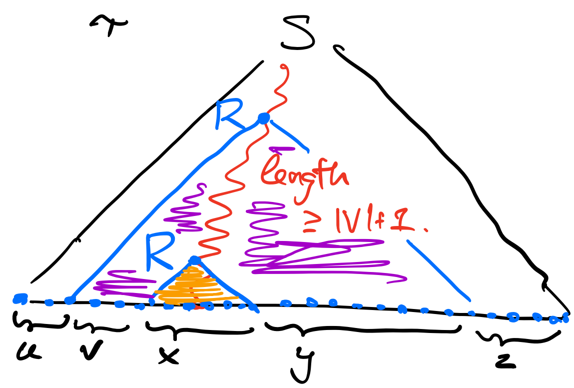
Let G be a CFG for some CFL A, and we'll let b denote the maximum number of symbols on the righthand side of any rule.



Thus any parse tree has at most b nodes at level 1, b^2 at level 2, b^h at level (or height) h, and so on.

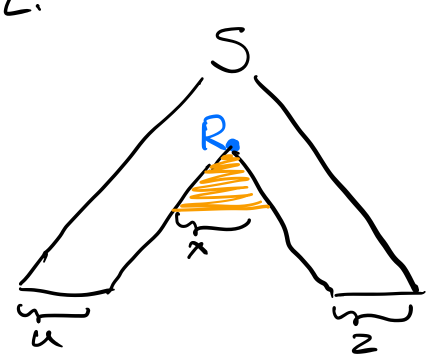
Set the pumping length $b^{|V|+1}$, where $|V|$ is the number of variables in G. Any string $s \in A$ of length at least $p = b^{|V|+1}$ must have parse tree(s) with height at least $|V|+1$. (Parse trees with height at most $|V|$ have $\leq b^{|V|}$ nodes.)

Let's let T be some parse tree for s with the smallest possible number of nodes. Because T has height at least $|V|+1$, there must be a path from the root of length $|V|+1$ (with $|V|+2$ variables.) Thus there is some variable R that appears twice in the lowest $|V|+1$ nodes.

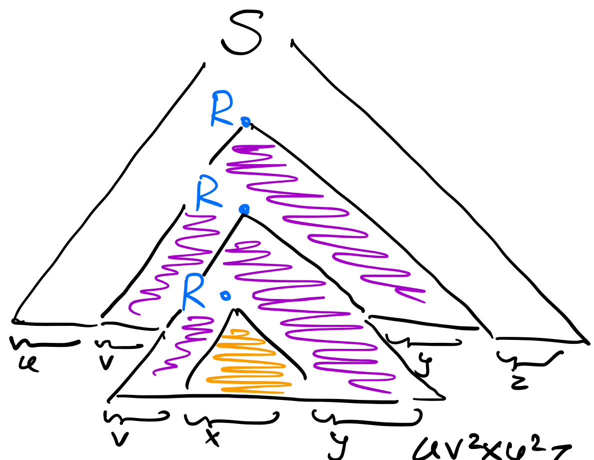


△ denotes the subtree derived from R. Divide s into u, v, x, y and z according to this picture.

We can substitute subtrees to make parse trees for new strings in L.



$uxz = uv^0xy^0z$

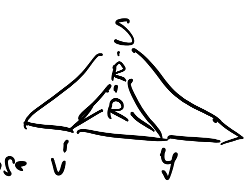


uv^2xy^2z

Remains to show that our conditions hold.

(1) $uv^ixy^iz \in L$ for $i \geq 0$. (Follows from pictures.)

(2) $|vy| > 0$. (If this were not true, then T would not be a subtree with the minimum number of nodes.)



(3) $|vxy| \leq p$. This is because R repeats in the bottom $|V|+1$ nodes of some path. So the height of the tree rooted at the first R is at most $|V|+1$, and it has at most $b^{|V|+1} = p$ leaves. ∎

<u>Example.</u> Using the PL for CF languages.

Goal: show $B = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

(1) Assume B is context-free. Thus B satisfies CFPL, and has a pumping length p. For all $s \in B$ with $|s| \geq p$, s can be divided into $uvxyz$ such that

$$uv^ixy^iz \in B \text{ for all } i \geq 0,$$
$$|vy| > 0,$$
$$|vxy| \leq p.$$

(2) Choose $s = a^p b^p c^p$, and consider all possible divisions, in 2 cases.

Case 1: v and y contain at most one type of symbol each. $\{a,b,c\}$

Thus, as $|vy| > 0$, vy contain at least one and at most two symbols from $\{a,b,c\}$. This means that pumping v and y to get uv^2xy^2z increases the number of some symbol in $\{a,b,c\}$ but does not increase some other symbol.

$$a\underbrace{aa}_{v}a\,b\underbrace{bb}_{y}b\,cccc \implies a\underbrace{aaaa}_{vv}a\,b\underbrace{bbbb}_{yy}b\,cccc$$

So $uv^2xy^2z \notin B$.

Case 2: At least one of v and y contains a's and b's or b's and c's. Now, pumping v and y gives us symbols out of order.

$$\cdots aa\underbrace{abb}_{v}b\cdots \qquad \cdots aa\underbrace{abb}_{v}\underbrace{abb}_{v}b\cdots$$

So $uv^2xy^2z \notin B$.

Thus any division of s fails the PL $\longrightarrow$ contradiction, so B is not CF.

Alternate logic.

$S = a^p b^p c^p$.

$$S = aa\cdots aaa\underbrace{b\cdots bbb}_{\text{length } p}ccc\cdots cccc$$

we know $|vxy| \le p$.

Thus either vxy has no a's or no c's.

Thus pumping v and y either doesn't increase a's or c's while increasing something else.

Example. Let $D = \{ww \mid w \in \{0,1\}^*\}$. We show D is not context-free.

(1) Assume for contradiction that D is CF. Thus D satisfies the CFPL and there exists some pumping length p such that for all strings $s \in D$, $|s| \ge p$, we have $s = uvxyz$ such that

$$uv^ixy^iz \in D \quad \forall i \ge 0,$$
$$|vy| > 0,$$
and $|vxy| \le p$.

(2) Choose $0^p 1 0^p 1$?

But – this string can be pumped.

$$\underbrace{0000\cdots0000}_{u}\ \underbrace{0}_{v}\ \underbrace{1}_{x}\ \underbrace{0}_{y}\ \underbrace{0000\cdots000\ 1}_{z}$$

We need to pick a new string. Why did this fail?

≈ v and y are repeating the same part of a string.

Choose $0^p 1^p 0^p 1^p$. $s \in D$, $|s| = 4p \ge p$.

We'll show every division of s into substrings fails one of our conditions.

Case 1: vxy is a substring of the first $0^p 1^p$ substring.

$$0000\cdots\underbrace{0000\,11\cdots}_{vxy}\,|\,111\ 0^p 1^p$$

Now, if we pump v and y, and consider uv^2xy^2z, the middle of our new string is now a 1 (because $|vxy| \le p$, $|vy| > 0$.)
(Increased the first part by 1 to p symbols)

Thus if we divide uv^2xy^2z in half, the first half starts with 0 and the second half starts with 1. So $uv^2xy^2z \notin D$.

<u>Case 2</u>: vxy is a substring of the last 0^p1^p substring
(Similar.)

<u>Case 3</u>: vxy straddles the midpoint of the string.

$$0^p 11\cdots \underbrace{111\,000}_{vxy}\cdots 0001^p.$$

Now: consider pumping down to the string $uv^0xy^0z = uxz$. We get the string $0^p1^i0^j1^p$ for some $i, j \le p$. At least one of $i, j < p$ because $|vy| > 0$. However, this means that $0^p1^i \ne 0^j1^p$, so $uv^0xy^0z \notin D$.

Thus no division of s satisfies our conditions, D fails the CFPL and is thus not context-free. ∎

Break. Back at 11:37.

---

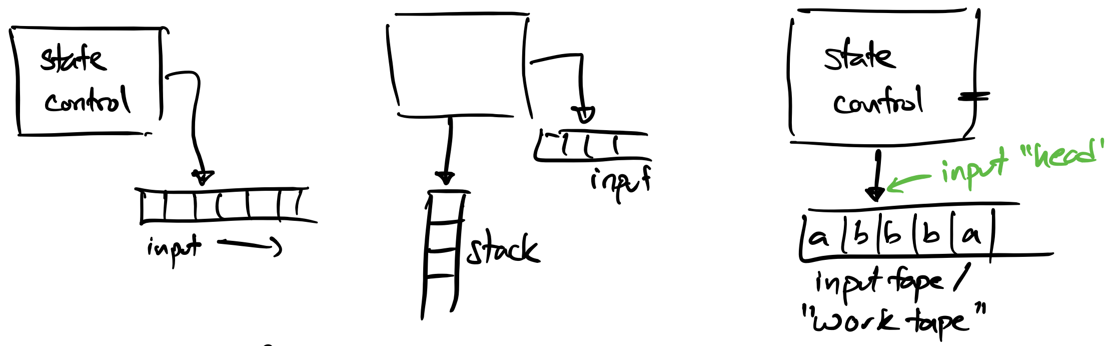
2. Turing Machines.

<u>Alan Turing</u>: (1912 - 1954)

- Invented the TM (1936)
- Invented the Turing Test. (can machines think?)
- Built a (sort of) computer during WWII, cracked the Enigma cipher.

Turing Machine: an automaton that can read and write on an infinite tape.

NFA/DFA: PDA: TM:



At every step of computation,

(1) read a symbol off the current tape square.

(2) enter a new internal state, write on the tape, and move tape head L or R.

Def. A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

where Q is a finite set of states,

Σ is the input alphabet,

Γ is the tape alphabet. $(\underline{\Sigma \subseteq \Gamma}, \sqcup \in \Gamma)$ blank symbol, space. fills rest of tape at beginning.

$q_0, q_{accept}, q_{reject}$ are start, accept, and reject states.

(deterministic!) (for now...) $\delta: \underbrace{Q \times \Gamma}_{\text{given a state and a tape symbol}} \longrightarrow \underbrace{Q}_{\text{next state}} \times \underbrace{\Gamma}_{\text{write symbol}} \times \underbrace{\{L, R\}}_{\text{move L or R.}}$.

How our TM computes:

(1) An input string $w = w_1 w_2 \cdots w_n \in \Sigma^*$ is written on the leftmost n squares of the tape.

(2) The "head" is on the leftmost tape square. The start state is q_0.

(3) Computation proceeds deterministically according to the transition function. (If we are in the leftmost square and go 'L', don't move.)

(state, read symbol) $\longrightarrow$ (state, write, L or R).

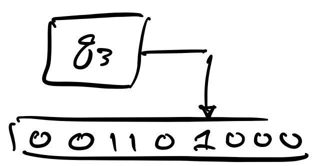
(4) Computation proceeds until we enter q_{accept} or q_{reject}, at which point we <u>immediately</u> stop and accept / reject.

// we might loop infinitely...

<u>Def.</u> A <u>configuration</u> of a TM is a snapshot of the machine at a particular step of computation: a state, tape contents, and the location of the tape head.

We write this uqv for a TM in state q, where u and v are the tape contents on either side of the head.

(Ex. $00110\,q_3\,1000$ corresponds to


)

Now we can say

uaq_ibv yields uq_jacv if $\delta(q_i, b) = (q_j, c, L)$

and similarly for moving R.

<u>Def.</u> A TM <u>accepts</u> a string w if there exists a sequence of configurations $C_1, C_2, \ldots, C_k$, such that

C_1 is the start configuration $q_0 w$, — in state q_0, w is on the tape, head is on w_1

C_i yields C_{i+1} for $i < k$,

and C_k is an accept configuration.

<u>Infinite Loops: Recognizability & Decidability</u>

Def. (Turing-recognizability.) The set of all strings on which a Turing Machine M accepts is the language $L(M)$ of M. A language is Turing-recognizable if some TM recognizes it.

(But: this doesn't imply we always reject strings not in the language — could loop.)

Def. (Turing-decidability.) A language L is (Turing)-decidable, if some TM decides it:

- accepts on all strings in L
- rejects on all strings not in L. (never loops).

(A TM that never loops a decider.)

Takeaway: decidable $\rightarrow$ recognizable.

Example. A TM that recognizes $A = \{0^{2^n} \mid n \geq 0\}$.

Implementation-level description. Describe how the TM manages the tape and moves the head, but doesn't formally specify the transition function.

M_2 = "On input w:

1. Read the input left to right, crossing off every other 0.
2. (Base case.) Accept if we just saw a single 0.
3. If the number of 0's was odd, reject,
4. If the number of 0's was even, return the head to the leftmost square and repeat from step 1."

$0^{2^3} = 00000000 \Rightarrow \cancel{0}0\cancel{0}0x0x0 \Rightarrow xxx0xxx0$

$\Rightarrow xxxxxxx0$

$\Rightarrow$ ✓ accept.

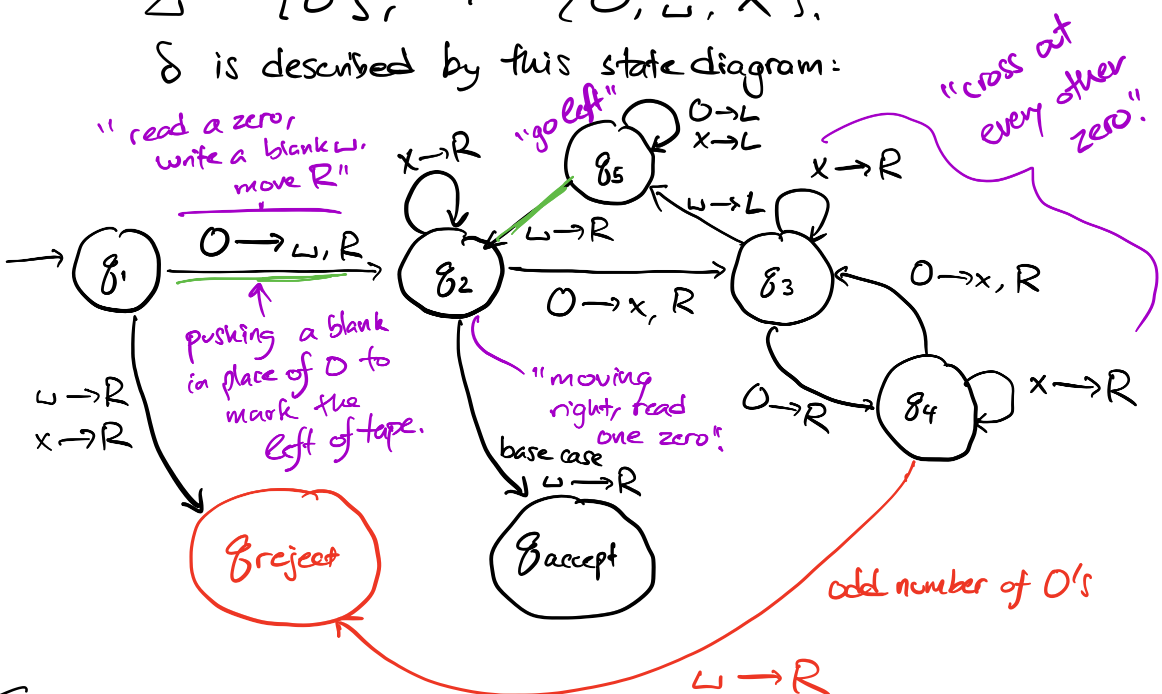
$0^6 = 000000 \Rightarrow x\underline{0}x\underline{0}x\underline{0} \Rightarrow xxx0xx$. ✗ reject.

Formal description of $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$.

$Q = \{q_1, q_2, \dots, q_5, q_{accept}, q_{reject}\}$

$\Sigma = \{0\}$, $\Gamma = \{0, \sqcup, x\}$.

δ is described by this state diagram:



Example on 0000

state	tape/head
q_1	0000
q_2	␣000
q_3	␣x00
q_4	␣x00
q_3	␣x0x␣

state	tape/head
q_5	␣x0x
q_5, q_5, q_5	␣x0x
q_2	␣x0x
q_2, q_3, q_3	␣xxx␣

q_3, q_5, q_5, q_5, q_2

⟵ to beginning

⟶ to end on x's

to q_{accept} from ␣xxx␣.