18.404 Recitation 6

Oct 9, 2020

Today's Topics

- Question: A_{TM} and a possibly infinite alphabet?
- Review: Recursion Theorem
  - Proving A_{TM} is undecidable with Recursion Theorem
- A_{LBA} is decidable
- E_{LBA} is undecidable
- ALL_{PDA} is undecidable
- $E_{2WAY\text{-}PDA}$ is undecidable
- Review: Closure Properties
- Review: Computability Class Relationships
- Review: Common Languages
- Review: Pumping Lemmas
- Bonus: $B = \{ 0^i 1^j 2^k \mid i,j,k \geq 0 \text{ and } i \geq k \text{ } \underline{\text{or}} \text{ } j \geq k \}$
  - B is a CFL
  - B is not a regular language

Question: A_{TM} and possibly infinite alphabet

- TM alphabets are always finite
- So then how does A_{TM} recognize <M,w> when languages of M may differ from A_{TM}?

- Trick: Encode alphabet of M into some fixed alphabet, say {0,1}
- Express k^{th} symbol of M's alphabet as 0^k1 for example

Review: Recursion Theorem

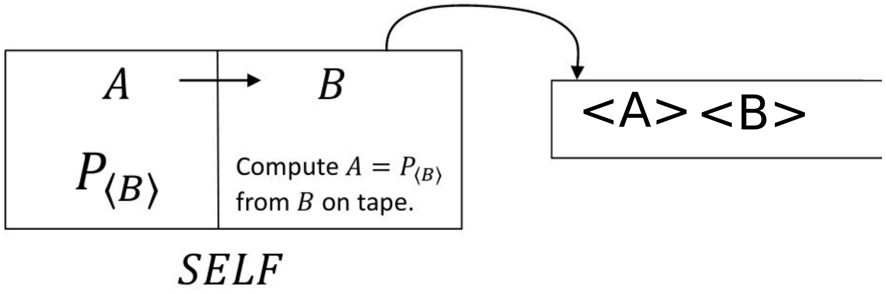
Goal is to show that a TM can retrieve its own description

- Define function $q: \Sigma^* \rightarrow \Sigma^*$ such that:
  - $q(w) = \langle P_w \rangle$
  - q on *w* returns description of TM that when run, prints *w* onto tape and halts.
  - Straightforward to see that such function can exist

Review: Recursion Theorem (cont.)

TM SELF has two parts, A and B

1. Run A = <P_B>
2. Run B = "1. Compute q(*tape contents*) to get <$P_{\textit{tape contents}}$> = <$P_B$> = A
   2. Prepend A to tape (currently has B) to get AB
   3. Halt with AB = <SELF> on tape"



Proving A_{TM} is Undecidable with Recursion Theorem

Proof by Contradiction: Assume some TM H decides A_{TM}

Consider TM R: (diagonalization argument)

R = “On input *w*

1. Get own description <R>
2. Use H on <R, *w*> to determine whether R accepts *w*
3. Do opposite of what H returns”

If H were to exist, then we could create such a TM R. But such TM R may never exist!

A_{LBA} is Decidable

LBA has bounded tape, so bounded number of state configurations

state configurations = $|Q| \cdot w \cdot |\Gamma|^w$

S = "on input <L, w>

1. Simulate L on w for |Q|*w*|Gamma|^w iterations
2. Accept if simulation accepts
   Reject if simulation rejects or is not yet halted"

E_{LBA} is Undecidable

Goal: To create a LBA that only accepts a valid comp. history of M on w

Reduce A_{TM} to E_{LBA} problem ($A_{TM} \leq_m E_{LBA}$)

Assume TM T decides E_{LBA}. Use T to create TM S which decides A_{TM}

S = "on input <M,w>

1. Create a LBA L which accepts only on comp. history M on w
   - Check if legal starting config for input w
   - Check if legal accepting config at the end
   - Check that all transitions Ci to Ci+1 are valid
2. Use TM T on L
3. If T accepts L, reject. Accept otherwise."

ALL_{PDA} is Undecidable

Goal: Create a PDA which accepts all non-valid comp. histories and all valid comp. histories EXCEPT M on w

Similar reduction and proof as for E_{LBA}. Use computation history method.

Assume TM T decides ALL_PDA.

S = "on input <M,w>

1. Create a P PDA which does not accept valid comp history M on w
   Non-det. branch to the following conditions:
   - Accept if starting config is NOT valid
   - Accept if ending config is NOT valid
   - Go to some point on tape. Push Ci onto the stack and compare against Ci+1. If NOT valid transition accept.
2. Use TM T on P. If T accept, then reject. Otherwise, accept."

ALL_{PDA} is Undecidable

Similar reduction and proof as for E_{LBA}. Use computation history method.

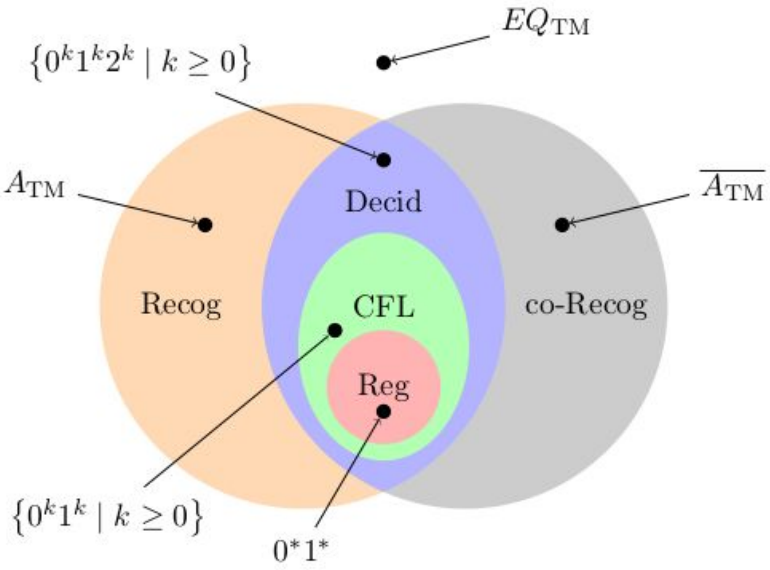
18.404 -> 404.81

$$\# \underbrace{\longrightarrow}_{C_1} \# \underbrace{\longleftarrow}_{C_2^R} \# \underbrace{\longrightarrow}_{C_3} \# \underbrace{\longleftarrow}_{C_4^R} \# \quad \cdots \quad \# \underbrace{\qquad\qquad}_{C_k} \#$$

Review: Closure Properties

Regular Languages	CFLs
Closed • Union • Concatenation • Kleene Star • Intersection • Negation	Closed • Union • Concatenation • Kleene Star
	Not-Closed • Intersection • Negation Note: CFL ∩ Reg. lang. = CFL

Review: Computability Class Relationships



Review: Common Languages

T-Decidable	T-Recognizable (undecidable)	T-coRecognizable (undecidable)	T-Unrecognizable (neither T-recog nor T-coRecog)
• A_{DFA} • A_{NFA} • E_{DFA} • EQ_{DFA} • A_{CFG} • E_{CFG}	• A_{TM} • negate(EQ_{CFG}) • $HALT_{TM}$	• E_{TM} • negate(A_{TM}) • EQ_{CFG}	• EQ_{TM} • negate(EQ_{TM})

Review: Pumping Lemmas (Regular Language)

For every regular language, there exists a pumping number $p \geq 1$ such that every string of length at least p can be written as w=xyz and satisfies:

- $|y| \geq 1$
- $|xy| \leq p$
- $(\forall n \geq 0)\ (xy^nz \in L)$

Review: Pumping Lemmas (CFLs)

For every CFL, there exists a pumping number $p \geq 1$ such that every string of length at least p can be written as $s = uvxyz$ and satisfies:

- $(\forall n \geq 0)\ (uv^n xy^n z \in L)$
- $|vy| \geq 1$
- $|vxy| \leq p$

Note: Need to make sure that all slide-windows in the string that CANNOT be pumped!