

Lecture 35: Undecidable language questions

CSCI 81
Spring, 2012

Kim Bruce

Rice's Theorem

- No non-trivial property of the SD languages is decidable
or equivalently
- Any language that can be described as $\{\langle M \rangle \mid P(L(M)) = \text{true}\}$ for any non-trivial property P , is not in D
- A property is *trivial* if it is either true for all languages or false for all languages.

Non-SD Languages

- Intuition: Usually involves infinite search or knowing TM will loop.
- Examples:
 - $\text{not-}H_{TM}$
 - $\{\langle M \rangle \mid L(M) = \Sigma^*\}$
 - $\{\langle M \rangle \mid M \text{ halts for no values}\}$

How to prove not SD?

- Show contradiction
- Show L is complement of SD/D language
- Reduction from known non-SD language

L is complement of SD/D language

- Let $H_{\neg ANY} = \{\langle M \rangle \mid M \text{ does not accept any string}\}$
- Complement: $\{\langle M \rangle \mid M \text{ accepts some string}\}$ is in SD/D
- Therefore $H_{\neg ANY}$ not in SD.

Alternate Proof using Reduction

- Show $\text{not-}H_{TM} \leq_M H_{\neg ANY}$
- Given $\langle M, w \rangle$, build M' s.t. M' erases tape and then runs M on w .
 - Then $\langle M, w \rangle \in \text{not-}H_{TM}$ iff $M' \in H_{\neg ANY}$
- Therefore $H_{\neg ANY}$ not in SD.

Contradiction

Skip for now!

- $TM_{Min} = \{\langle M \rangle \mid M \text{ is minimal generating } L(M)\}$
- Proof by contradiction:
 - Suppose TM_{Min} in SD. Then exists TM Enum that enumerates the elts of TM_{Min}
 - Define M' to operate as follows:
 - Run Enum until generates an M' s.t.

Decision Problems for Regular Languages

- For FSMs/Regular languages, most things decidable:
 - $A_{FSM} = \{\langle M, w \rangle \mid M \text{ is an FSM and } w \in L(M)\}$
 - $E_{FSM} = \{M \mid M \text{ is a FSM and } L(M) = \emptyset\}$
 - $TOTAL_{FSM} = \{M \mid M \text{ is a FSM and } L(M) = \Sigma^*\}$
 - $EQUAL_{FSM} = \{\langle M, N \rangle \mid M \text{ is a FSM and } L(M) = \Sigma^*\}$

Decision Problems for CFGs

- Not for PDAs/CFGs:
 - $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG and } w \in L(G) \}$
 - $E_{CFG} = \{ G \mid G \text{ is a CFG and } L(G) = \emptyset \}$
 - $Finite_{CFG} = \{ G \mid G \text{ is a CFG and } L(G) \text{ is finite} \}$
 - $TOTAL_{CFG} = \{ G \mid G \text{ is a CFG and } L(G) = \Sigma^* \}$
 - $EQUAL_{CFG} = \{ \langle G, G' \rangle \mid G, G' \text{ are CFGs and } L(G) = L(G') \}$
 -
- Not Decidable!*
-

$TOTAL_{CFG}$ is the key!

- Assume $TOTAL_{CFG}$ is undecidable and show others undecidable.
- $EQUAL_{CFG} = \{ \langle G, G' \rangle \mid G, G' \text{ are CFGs \& } L(G) = L(G') \}$
 - Let G_{Tot} be grammar s.t. $L(G_{Tot}) = \Sigma^*$
 - Suppose *Oracle* decides $EQUAL_{CFG}$
 - To decide if G total, ask oracle about $\langle G, G_{Tot} \rangle$.
 - If yes, then G is total. If no, then not.
 - By contradiction, $EQUAL_{CFG}$ is undecidable

Minimizing PDA's

- $MIN_{PDA} = \{ \langle M_1, M_2 \rangle : M_2 \text{ is a minimization of } M_1 \}$ is undecidable.
 - Proof: Suppose Oracle to solve MIN_{PDA} .
Let P_a be PDA with one state that accepts everything (never push anything on stack).
 - Given cfg G , construct equivalent PDA P s.t. $L(P) = L(G)$.
 - Submit $\langle P, P_a \rangle$ to Oracle and get answer to $L(P) = L(G) = \Sigma^*$

Other Undecidable

- Is $L(G)$ inherently ambiguous?
- Is $L(G) \cap L(G') = \emptyset$?
- If $L(G) \subseteq L(G')$?
- Is complement of $L(G)$ a cfl?
- Is $L(G)$ regular?

Total_{CFG} is Undecidable

- Recall: Configuration of TM M is a 4 tuple:
 - M 's current state
 - nonblank portion of the tape before the read head,
 - the character under the read head,
 - the nonblank portion of the tape after the read head

Computation

- A computation of M is a sequence of configurations:
 C_0, C_1, \dots, C_n for some $n \geq 0$ such that:
 - C_0 is the initial configuration of M ,
 - C_n is a halting configuration of M , and:
 - $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \dots \vdash_M C_n$.
- Computation history is sequence of configurations.

Proof

- Theorem: Total_{CFG} is undecidable
 - Proof: Reduction via halting
 - Given M, w , build grammar G generating language L composed of all strings in Σ^* except any representing a (halting) computation history of M on w .
 - Suppose *Oracle* solves Total_{CFG}. Run on G .
 - If says yes, then M doesn't halt on w
 - If says no, then exist halting computation from w .
 - Contradiction!

Recognizing Computation Histories

- Build PDA rather than CFG and then convert.
- For s to be computation history of M on w :
 - It must be a syntactically valid computation history.
 - C_0 must correspond to M being in its start state, with w on the tape, and with the read head to the left of w .
 - The last configuration must be a halting configuration.
 - Each configuration after C_0 must be derivable from the previous one according to the rules in δ_M .

Invalid Computation Histories

- PDA can guess one of the following fails (use non-determinism!)
 - Invalid syntax for computation.
 - C_0 not rep. opening config (bad state or input)
 - Last configuration not halting
 - Successor config not follow from previous according to transition function.

Recognizing Invalid Computations

- Last can be done easily if have extra tape on TM (or extra read head)
- To check last point (transitions incorrect) with pda, must save a configuration on stack in order to check next.
- But elements popped off stack in opposite order added (LIFO). How to compare??

Boustrophedon??

- Solve by writing every other configuration backwards, so can compare via stack.
 - This text is written
 - ot yaw yzarc siht ni
 - show Boustrophedon style.
- Assume computation history written in Boustrophedon style
- Exists iff regular history exists!

Invalid Computation History

- If guessing particular step of computation is wrong in $C_0 C_1^R C_2 \dots$
 - Keep track of which direction going
 - Push C_i (possibly reversed) onto stack
 - Compare C_{i+1} to make sure that there is an error
 - In copying unchanged portion of configuration or
 - In changing part reflecting transition.
- Hence whether $L(G) = \Sigma^*$ is undecidable.