Context-Sensitive Languages
Wednesday, March 30, 2022
Announcements

• HW 8 due 4/3 11:59pm
Last Time: Rice’s Theorem

All languages that look like:

\[ \text{ANYTHING}_{TM} = \{ <M> \mid M \text{ is a TM and “… anything …” about } L(M) \} \]

... are **undecidable**!

This means: There’s **no algorithm** to compute anything about the **language** (behavior) of **TMs**, i.e., about programs!
Last Time:

- \( A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \)
- \( A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \)
- \( A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \)
- \( E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \} \)
- \( E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \} \)
- \( E_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \)
- \( E_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)
- \( E_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \)
- \( E_{\text{TM}} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \)

Langs about non-TMs: Decidable
Langs about TMs: Undecidable

Decidable
Decidable
Undecidable
Decidable
Decidable
Undecidable
Decidable
Undecidable

It breaks the pattern?
Last Time:

\[ A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \} \]
\[ A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \]
\[ A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \]

Decidable
Decidable
Undecidable

• In hindsight, it makes sense that:
  • A restricted TM (a decider) ...
  • ... can’t simulate an unrestricted TM (a recognizer)

• But what about:
  • A restricted TM (a decider) ...
  • ... simulating an even more restricted TM (a ???)
Context-Sensitive Languages

- context-sensitive languages (CSL)
- generated by: context-sensitive grammars (CSG)
- recognized by: linear bounded automata (LBA)

---

Chomsky hierarchy

From Wikipedia, the free encyclopedia

In formal language theory, computer science, and linguistics, the Chomsky hierarchy (also referred to as the Chomsky–Schützenberger hierarchy) is a containment hierarchy of classes of formal grammars.
A linear bounded automaton is a restricted type of Turing machine wherein the tape head isn’t permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is—in the same way that the head will not move off the left-hand end of an ordinary Turing machine’s tape.
Theorem: \( A_{\text{LBA}} \) is decidable

\[ A_{\text{LBA}} = \{ \langle M, w \rangle | M \text{ is an \text{LBA} that accepts string } w \} \]
Flashback: TM Configuration = State + Head + Tape

Textual representation of “configuration”

1st char after state is current head position
How Many Possible Configurations ...

• Does an LBA have?
  • $q$ states
  • $g$ tape alphabet chars
  • tape of length $n$

• # of possible ways to fill the tape:
  • $g^n$

• # of possible head positions:
  • $n$

• Total Possible Configurations = $qng^n$
Theorem: $A_{LBA}$ is decidable

$A_{LBA} = \{\langle M, w \rangle | M \text{ is an LBA that accepts string } w \}$

Proof: Create decider for $A_{LBA}$

On input $\langle M, w \rangle$:

- Simulate $M$ on $w$:
  - If $M$ accepts $w$, then accept $\langle M, w \rangle$
  - If $M$ rejects $w$, then reject $\langle M, w \rangle$
- If $M$ runs for more than $qng^n$ steps ... 
- ... then we are in a loop so halt and reject!

Termination argument?
Theorem: $E_{LBA}$ is undecidable

$E_{LBA} = \{ \langle M \rangle \mid M \text{ is an LBA where } L(M) = \emptyset \}$
Flashback: TM Configuration Sequences

\[ M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}) \]

**Single-step**

(Left)

\[ \alpha b q_1 a \beta \vdash \alpha q_2 b x \beta \]

if \( \delta(q_1, a) = (q_2, x, L) \)

(Right)

\[ \alpha q_1 a \beta \vdash \alpha x q_2 \beta \]

if \( q_1, q_2 \in Q \)

\[ \delta(q_1, a) = (q_2, x, R) \]

\( a, x \in \Gamma \quad \alpha, \beta \in \Gamma^* \)

**Extended**

- **Base Case**
  
  \[ I \vdash^* I \text{ for any ID } I \]

- **Recursive Case**
  
  \[ I \vdash^* J \text{ if there exists some ID } K \]
  
  such that \( I \vdash K \) and \( K \vdash^* J \)
Theorem: $E_{LBA}$ is undecidable

Proof, by contradiction:

• **Assume** $E_{LBA}$ has decider $R$; use to create decider for $A_{TM}$:

  • **On input** $<M, w>$, where $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:
    • Construct LBA $B$, whose input is a sequence of $M$ configs:
      • $B$ accepts sequences of $M$ configurations where $M$ accepts $w$, i.e.,
        • First configuration is $q_0w_1w_2 \cdots w_n$
        • Last configuration has state $q_{accept}$
        • Each pair of adjacent configs is valid according to $M$’s $\delta$
    • Run $R$ with $B$ as input:
      • If $R$ accepts $B$, then $B$’s language is empty
        • So $M$ has no sequence of configs that accepts $w$; so reject!
      • If $R$ rejects $B$, then $B$’s language is not empty
        • So $M$ has a sequence of configs that accepts $w$; so accept!

\[ E_{LBA} = \{ \langle M \rangle \mid M \text{ is an LBA where } L(M) = \emptyset \} \]
Theorem: $\text{ALL}_{\text{CFG}}$ is undecidable

$\text{ALL}_{\text{CFG}} = \{ \langle G \rangle | G \text{ is a CFG and } L(G) = \Sigma^* \}$

Proof, by contradiction

• Assume $\text{ALL}_{\text{CFG}}$ has a decider $R$. Use it to create decider for $A_{\text{TM}}$:

  On input $<M, w>$:
  
  • Construct a PDA $P$ that rejects sequences of $M$ configs that accept $w$
  • Convert $P$ to a CFG $G$ (previous class)
  • Give $G$ to $R$:
    • If $R$ accepts, then $M$ has no accepting config sequences for $w$, so reject
    • If $R$ rejects, then $M$ has an accepting config sequence for $w$, so accept
A PDA That Rejects TM $M$ Config Sequences

On input # $\ldots$ #, nondeterministically:

- Reject if $C_1$ is not $q_0 w_1 w_2 \ldots w_n$
- Reject if $C_i$ does not have $q_{accept}$
- Reject if any $C_i$ and $C_{i+1}$ is invalid according to $\delta$:
  1. Push $C_i$ onto the stack
  2. Compare $C_i$ with $C_{i+1}$ (reversed):
     a) Check that initial chars match
     b) On first non-matching char:
        • check that next 3 chars is valid according to $\delta$
        • Each possible $\delta$ can be hard-coded since $\delta$ is finite
     c) Continue checking remaining chars
     d) Reject whenever anything is invalid

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$
Algorithms For CFLs

- $A_{CFG} = \{ \langle G, w \rangle | G \text{ is a CFG that generates string } w \}$
  - Decidable

- $E_{CFG} = \{ \langle G \rangle | G \text{ is a CFG and } L(G) = \emptyset \}$
  - Already proved this is decidable

- $ALL_{CFG} = \{ \langle G \rangle | G \text{ is a CFG and } L(G) = \Sigma^* \}$
  - Just proved this is undecidable

These two languages tell us something about the threshold of decidability.
Exploring the Limits of CFLs

- **A CFL:** \( \{ w_1 \# w_2 \mid w_1 \neq w_2 \} \)
  - PDA nondeterministically checks matching positions in 1st/2nd parts
  - And rejects if **any** are not the same
  - I.e., Each computation branch is independent, i.e. “context free”

- **Not a CFL:** \( \{ w_1 \# w_2 \mid w_1 = w_2 \} \)
  - Can nondeterministically check matching positions
  - But needs to accept only if **all** branches match
  - I.e., Each branch is **not independent**

Summary: the “context freeness” of CFLs has to do with dependency between non-deterministic computation branches (This is also why **union** is closed for CFLs but **intersection** is not)

This is similar to the config-rejecting PDA

This is similar to the **ww** language (not pumpable)

An config-accepting PDA would be like this language ... i.e., not a CFL!
Algorithms For CFLs

• $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$
  Decidable

• $E_{CFG} = \{ \langle G' \rangle \mid G \text{ is a CFG and } L(G') = \emptyset \}$
  Already proved this is decidable

• $ALL_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \}$
  Just proved this is undecidable

• $EQ_{CFG} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$
  Undecidable

(Still need to prove this is undecidable)
Theorem: $EQ_{CFG}$ is undecidable

$EQ_{CFG} = \{(G, H) | G$ and $H$ are CFGs and $L(G) = L(H)\}$

- Proof by contradiction: Assume $EQ_{CFG}$ has a decider $R$
- Use $R$ to create a decider for $ALL_{CFG}$:

On input $\langle G \rangle$:

- Construct a CFG $G_{ALL}$ which generates all possible strings
- Run $R$ with $G$ and $G_{ALL}$
- Accept $G$ if $R$ accepts, else reject
The Post Correspondence Problem (PCP)
A unique undecidable problem
A Non-Formal Languages Undecidable Problem: $PCP$

• Let $P$ be a set of "dominos" $\{ [\frac{t_1}{b_1}], [\frac{t_2}{b_2}], \ldots, [\frac{t_k}{b_k}] \}$
  • Where each $t_i$ and $b_i$ are strings

• E.g., $P = \left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}$

• A match is:
  • A sequence of dominos with the same top and bottom strings

• E.g., $\left[ \frac{a}{ab} \right] \left[ \frac{b}{ca} \right] \left[ \frac{ca}{a} \right] \left[ \frac{a}{ab} \right] \left[ \frac{abc}{c} \right] \rightarrow \begin{array}{cccccc}
\text{a} & \text{b} & \text{c} & \text{a} & \text{a} & \text{b} \\
\text{a} & \text{b} & \text{c} & \text{a} & \text{a} & \text{b} \\
\end{array}$

• Then: $PCP = \{ <P> \mid P \text{ is a set of dominos with a match} \}$
Theorem: \( \text{PCP} \) is undecidable

\[ \text{PCP} = \{ \langle P \rangle \mid P \text{ is a set of dominos with a match} \} \]

Proof by contradiction:

Assume \( \text{PCP} \) is decidable, has decider \( R \); use it to create decider for \( A_{\text{TM}} \):

On input \( \langle M, w \rangle \):

1. Construct a set of dominos \( P \) that has a **match** only when \( M \) accepts \( w \)
2. Run \( R \) with \( P \) as input
3. Accept if \( R \) accepts, else reject

The trick: \( P \) has \( M \)'s TM configurations as its domino strings

So a match is a sequence of configs showing \( M \) accepting \( w \)!
PCP Dominos

• First domino: \[
\begin{array}{c}
\# \\
\#q_0w_1w_2\cdots w_n\#
\end{array}
\]

• **Key idea:** add dominos representing valid TM steps:

  if \( \delta(q, a) = (r, b, R) \), put \[\frac{q_a}{b_r} \] into \( P \)

  if \( \delta(q, a) = (r, b, L) \), put \[\frac{c_qa}{rcb} \] into \( P \)

• For the tape cells that don’t change: put \[\frac{a}{a} \] into \( P \)

• Top can only “catch up” if there is an accepting config sequence
PCP Example

• Let $w = 0100$ and $\delta(q_0, 0) = (q_7, 2, R)$ so $\begin{bmatrix} q_0 & 0 \\ 2q_7 \end{bmatrix}$ in $P$
PCP Dominos (accepting)

- When accept state reached, let top “catch” up:

  For every \( a \in \Gamma \),

  \[
  \frac{a}{q_{\text{accept}}} \quad \text{and} \quad \frac{q_{\text{accept}}}{a}
  \]

  put into \( P \)

  Only possible match: **accepting sequence of TM configs**

  Bottom “eats” one char

  “eat” one char
Check-in Quiz 3/30

On gradescope