Complement of context-free languages are not context-free.

Consider the language $L = \{ww^R : w \in \{0,1\}^*\}$.

- $L \neq \emptyset$, as $ww^R$ is even for $w = \epsilon$.
- $L \neq \{\epsilon\}$, as $ww^R$ is odd for $w = \epsilon$.
- $L \neq \Sigma^*$, as $ww^R$ is even for $w = \emptyset$.

Consider the language $L = \{a^n b^n c^n : n \geq 0\}$.

- $L \neq \emptyset$, as $a^2 b^2 c^2$ is in the language.
- $L \neq \{\epsilon\}$, as $a^\epsilon b^\epsilon c^\epsilon$ is not in the language.
- $L \neq \Sigma^*$, as $a^\emptyset b^\emptyset c^\emptyset$ is not in the language.

Similarly, state that the class of context-free languages is not closed under complementation.

Theorem: The class of context-free languages is not closed under complementation.
Un-recognizable languages

- $A_{TM}$ is Turing recognizable
- (theorem 4.22p181) A language $L$ is decidable if and only if $L$ and its complement $\overline{L}$ are both Turing recognizable
  - $L$ is called co-Turing-recognizable if its complement $\overline{L}$ is Turing-recognizable
- (corollary 4.23p182) $\overline{H}_{TM}$ and $\overline{A}_{TM}$, the complements of the halting problem and the acceptance problem, are not Turing recognizable

Un-recognizable languages

- $\overline{A}_{TM}$ is not Turing-recognizable but it is co-Turing-recognizable
- $E_{TM} = \{ G \mid G$ is a TM and $L(G) = \emptyset \}$ is not Turing recognizable
- $EQ_{TM} = \{ < G, H > \mid$ TMs $G, H$ and $L(G) = L(H) \}$ is not even co-TM-recognizable

Classifying problems

- Recognizable (RE)
  - Decidable (Recursive)
  - Context free
  - Regular
- $A_{TM}$ (co-RE)

The halting problem again

- Reductions between two problems "A reduces to B"
- $A \rightarrow B$, if we can solve $B$ then we can solve $A$
- (theorem 5.1p188) The halting problem is undecidable
- $HALT_{TM} = \{ < M, w > \mid M$ halts on input $w \}$
  - prove by reducing the acceptance problem $A_{TM}$ to $HALT_{TM}$
- Assume $R$ solves $HALT_{TM}
Reducibility and undecidability

The halting problem again

- Construct $S$ as follows:
  - $S$: On input $< M, w >$
    1. Run $R(< M, w >)$
    2. if $R$ rejects, reject
    3. if $R$ accepts, simulate $M$ on $w$
- $S$ solves $A_{TM}$ if $R$ exists

Reductions

- Proving undecidability using reductions
- To prove $L$ is undecidable, prove the following:
  "if $L$ is decidable then $A_{TM}$ is decidable"
- Proofs by contradiction based on a construction
- Given a TM for $L$ show how you can use it to solve $A_{TM}$ or any other undecidable problem

Emptiness

- (theorem 5.2p189) The emptiness problem for TMs is undecidable
- $E_{TM} = \{ < M > \mid L(M) = \emptyset \}$
- Proof: reduce from $A_{TM}$
  - Given a TM $R$ for $E_{TM}$ build one for $A_{TM}$
  - Modify $M$ to accept only $w$ if it is non-empty
- $M_1$: on input $< x >$
  1. if $x \neq w$, reject
  2. if $x = w$ simulate $M$ on $w$ and accept if $M(w)$ accepts

- $S$: on input $< M, w >$
  1. From $M$ construct $M_1$
  2. Simulate $R$ on $M_1$
- $S$ constructs $M_1$ by adding some new states to it that simply check for $x = w$
  
  $L(M_1) = \emptyset$ \hspace{1cm} $M$ does not accept $w$
  $L(M_1) = \{ w \}$ \hspace{1cm} $M$ accepts $w$
Regularly

- (Theorem 5.3p191) $\text{REGULAR}_{TM}$ is undecidable
- $\text{REGULAR}_{TM} = \{ < M > | L(M) \text{ is regular} \}$
- Proof: reduce from $A_{TM}$
- Assume $\text{REG}(M)$ decides regularity.
- Use it to solve $A_{TM}(M, w)$
- $S$: on input $< M, w >$
  - Construct $M_2$: on input $x$
    - if $x$ is $0^n1^n$ accept
    - otherwise run $M(w)$ and accept if it accepts
  - Run $\text{REG}$ on $M_2$
  - Accept if $\text{REG}(M_2)$ accepts

Other undecidable properties

- Acceptance, emptiness, regularity, equivalence
- Is $L(M)$
  - Context free
  - Finite
  - $\Sigma^*$
Rice’s Theorem

- Rice’s theorem: all non-trivial TM language properties are undecidable
- important to note: Rice’s theorem refers to properties of languages; not of the Turing machines

Let $T_\emptyset$ be a TM that always rejects $L(T_\emptyset) = \emptyset$

wlog assume $T_\emptyset \not\in P$ (otherwise proceed with $\overline{P}$)

$P$ non-trivial implies $\exists T \in P$

use the ability of $P()$ to distinguish between $T_\emptyset$ and $T$

Rice’s Theorem

- (problem 5.28p213) Let $P$ be a non-trivial property of the language of a TM. Prove that determining if the language of a given TM has this property is undecidable
  
  $P(M) = \{ < M > \mid L(M) \text{ has property } P \}$

  - non-trivial property: it contains some but not all TM languages
  - $P$ is a property of the TM language whenever $L(M_1) = L(M_2)$ then $M_1 \in P$ iff $M_2 \in P$

  Proof: assume $P()$ decides the property $P$

  reduce $A_{TM}$ to $P()$ by constructing $S$

- define $S$: on input $< M, w >$
  
  - Based on $M$ and $w$ construct $M_w(x)$
    
    - Simulate $M(w)$ if it halts and rejects, reject
    
    - if it accepts, simulate $T$ on $x$. if it accepts, accept
  
  - Use $P()$ to determine whether $M_w$ has property $P$. if it does, accept, otherwise reject

  $M_w$ simulates $T$ if $M(w)$ accepts

  - $L(M_w) = L(T)$ if $M$ accepts $w$
  
  - $L(M_w) = \emptyset$ otherwise
Reductions

- Reductions so far have been straightforward
- Properties that involve TM
- For more general questions, use different reductions

Computation histories

(definition 5.5p193) An accepting computation history of $M$ on $w$ is a sequence of configurations $C_1, \ldots, C_k$
- $C_1$ is the start configuration of $M$ on $w$
- $C_k$ is an accepting configuration
- $C_i \rightarrow_M C_{i+1}$, a valid computation stem of $M$
- If $C_k$ is a rejecting configuration, then this sequence is a rejecting computation history
- Computation histories are finite sequences
- If $M$ does not halt on $w$ no computation sequence exists

Deciding CF properties

(theorem 5.13p197) $ALL_{CFL}$ is undecidable
- $ALL_{CFL} = \{ < G > | G$ is a CFG and generates all $\Sigma^* \}$
- Proof: use computation histories to reduce $A_{TM}$ to this problem
- $A_{TM}$: Does $M$ accept $w$?
- $A_{TM}$: Does there exist an accepting computation history for $M$ on $w$?
CFL

- \( < M, w > \notin A_{TM} \) implies all histories \( x \) are non-accepting
- Define \( L_{CFL} \) a CFL that contains all accepting histories \( x \) of \( M \) on \( w \)
- \( < M, w > \notin A_{TM} \) translates to \( x \in L_{CFL} \)?
- a CFG \( G \) can describe all accepting histories
- \( x = (C_1, \ldots, C_k) \in G \) iff
  1. \( C_1 \) is not proper start configuration, or
  2. there is an invalid \( C_i \rightarrow C_{i+1} \)
  3. last configuration in \( x \) is not accepting

\( \text{CFL equality} \)

- theorem: \( EQ_{CFL} = \{ < G_1, G_2 > | G_1, G_2 \text{ are CFG and } L(G_1) = L(G_2) \} \)
- proof: let \( G' = \Sigma^* \). decide \( < G > \) is in \( ALL_{CFL} \) by \( EQ(G, G') \)

Post correspondence problem

- An undecidable problem concerning manipulations of strings
- a domino contains two strings (top and bottom)
  \[
  \begin{bmatrix}
  a \\
  bc
  \end{bmatrix}
  \]
- a collection of dominos
  \[
  \left\{ \begin{bmatrix} b \\ ca \end{bmatrix}, \begin{bmatrix} a \\ ab \end{bmatrix}, \begin{bmatrix} ca \\ a \end{bmatrix}, \begin{bmatrix} abc \\ c \end{bmatrix} \right\}
  \]
- PCP: given a collection of dominos find a match, a sequence (repetitions allowed) such that the top string is the same as the bottom
Post correspondence problem

- a collection of dominos
  \[
  \left\{ \left[ \frac{b}{ca} \right], \left[ \frac{a}{ab} \right], \left[ \frac{ca}{a} \right], \left[ \frac{abc}{c} \right] \right\}
  \]
- PCP: given a collection of dominos find a match, a sequence (repetitions allowed) such that the top string is the same as the bottom
  \[
  \left[ \frac{a}{ab} \right] \left[ \frac{b}{ca} \right] \left[ \frac{ca}{a} \right] \left[ \frac{a}{ab} \right] \left[ \frac{abc}{c} \right]
  \]

PCP is undecidable advanced topic

- (theorem 5.15p200) PCP is undecidable
- proof: reduce from \(A_{TM}\) using computation histories
- given \(M\) and \(w\), construct the dominos to be parts of the computation history of \(M\) on \(w\)
- these dominos will form a match only of an accepting computation history exists
- start with \(MPCP\), the modified PCP: the match must start with the first domino \(\left[ \frac{t_1}{b_1} \right]\)

PCP is undecidable advanced topic

- for some collections, a match does not exist
- Post correspondence problem: determine whether a given collection of dominos has a match
- collection: \(P = \left\{ \left[ \frac{t_1}{b_1} \right], \left[ \frac{t_2}{b_2} \right], \ldots, \left[ \frac{t_k}{b_k} \right] \right\}\)
- a match is a sequence \(i_1, i_2, \ldots, i_n\) such that
  \[t_1 t_2 \cdots t_n = b_1 b_2 \cdots b_n\]
- the PCP language is
  \[PCP = \{ \langle P \rangle | \text{a PCP problem that has a match} \}\]

- \(#q_0 w_1 w_2 \cdots w_n\#\) is the starting configuration of \(M\) on \(w\)
- put \(\left[ \begin{array}{c} \#q_0 w_1 w_2 \cdots w_n \# \\ \# \end{array} \right]\) as the first domino
- need to start from this domino- need to match the bottom part at the top
- this forces the next domino to have a certain kind of top part
- that domino will have a bottom part that is the next configuration of \(M(w)\)
PCP is undecidable advanced topic

<table>
<thead>
<tr>
<th>$\delta(q, a) = (r, b, R)$</th>
<th>$\delta(q, a) = (r, b, L)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q a \rightarrow b \delta$</td>
<td>$q a \rightarrow b \gamma$</td>
</tr>
<tr>
<td>$\forall a \in \Gamma$</td>
<td>$\forall a \in \Gamma$</td>
</tr>
<tr>
<td>copy #, add a blank</td>
<td>$a \text{accept}$</td>
</tr>
<tr>
<td>$\gamma \text{accept}$</td>
<td>$\gamma \text{accept}$</td>
</tr>
</tbody>
</table>

last step: build in the problem the requirement to start with the first domino
- for $u = u_1 u_2 \cdots u_n$ define
  - $u_\star = u_1 \star u_2 \star \cdots \star u_n$
  - $u_* = u_1 \ast u_2 \ast \cdots \ast u_n$
- convert $\{ [t_1 b_1], [t_2 b_2], \ldots, [t_k b_k] \}$
- to $\{ [t_1 \ast b_1 \star], [t_1 \ast b_1 \ast], [t_2 \ast b_2 \ast], \ldots, [t_k \ast b_k \ast], [\ast \ast \ast] \}$

PCP is undecidable advanced topic

- Many different kinds of reductions
- Mapping or many-one reducibility
- (definition 5.17p206) a function $f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if some TM on input $w$ halts with $f(w)$ on its tape
- (definition 5.20p207) $A$ is mapping reducible to $B$ written $A \leq_m B$ if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that for all $w$
  $$w \in A \iff f(w) \in B$$
  The function $f$ is called the reduction from $A$ to $B$

- if $A \leq_m B$ and $B$ is decidable then $A$ is decidable
- $A \leq_m B$ if $A$ is undecidable then $B$ is undecidable
- if $A \leq_m B$ and $B$ is Turing recognizable then $A$ is T-recognizable
- if $A \leq_m B$ and $A$ is not T-recognizable then $B$ is not T-recognizable
- $A \leq_m B$ is the same as $\overline{A} \leq_m \overline{B}$
Equality of TM languages advanced topic

- (theorem 5.30p210) $EQ_{TM}$ is neither T-recognizable nor co-T-recognizable
- prove two things:
  
  \[
  \begin{array}{c}
  EQ_{TM} \text{ not T-recognizable} \\
  EQ_{TM} \text{ not co-T-recognizable} \\
  EQ_{TM} \text{ not T-recognizable}
  \end{array}
  \]
  
  $A_{TM} \leq_{m} EQ_{TM}$

$A_{TM} \leq_{m} EQ_{TM}$ advanced topic

- $A_{TM}$ reduces to $\overline{EQ}_{TM}$
- $F$: on input $M, w$
  1. construct two machines
     - $M_1 = \text{reject all inputs}$
     - $M_2 = \text{run } M(w) \text{ if it accepts, accept}$
  2. output $< M_1, M_2 >$
  
  \[
  \begin{array}{c}
  M(w) \text{ accepts} \\
  M(w) \text{ does not accept}
  \end{array}
  \]
  
  \[
  \begin{array}{c}
  M_1 \text{ nothing, } M_2 \text{ everything} \\
  M_1 \text{ nothing, } M_2 \text{ nothing}
  \end{array}
  \]