Does an LBA accept no strings?

Let $E_{LBA} = \{ \langle M, w \rangle \mid M$ is an LBA and $L(M) = \emptyset \}$.

Theorem: $E_{LBA}$ is undecidable.

Proof idea:
- Assume $E_{LBA}$ is decidable and do a reduction from $A_{TM}$.
- For a TM $M$ and input $w$.
  - Is there an LBA $B$ that accepts some string when $M$ accepts $w$?
  - Is $L(B)$ the same as the set of accepting computation histories for $M$ on $w$?
  - But does $B$ exist? An LBA that accepts accepting computations of a TM?

Moreover: $E_{LBA}$ is undecidable.

$\{ \emptyset = \langle M, w \rangle \mid M \text{ is an LBA and } L(M) = \emptyset \}$ is $E_{LBA}$.

Overview

Matching Reducibility

Post Correspondence Problem

Reductions via Computational Histories
Proof

**Does** $\mathcal{E}_{\text{LBA}}$ **decide** $\mathcal{A}_{\text{TM}}$?

- $\mathcal{E}_{\text{LBA}}$ is decidable, so $\mathcal{A}_{\text{TM}}$ is decidable.

$\mathcal{S}$ **accepts** $\langle M, w \rangle$ exactly when $\mathcal{R}$ **rejects** $\langle B \rangle$ which is exactly when $\mathcal{M}$ has an accepting computation of $w$. So is $\mathcal{A}_{\text{TM}}$ decidable. Contradiction. So, our assumption is false.

**How to construct the LBA $B$**

- **Accepting computation** $C_1, C_2, ..., C_l$
  - Written on tape as single string, with configurations separated by the # symbol

  - $B$ will determine whether $C_1$ is the start configuration for $M$ on $w$ and $B$ is a decider on previous slide

  - $S$ = On input $\langle M, w \rangle$ where $M$ is a TM and $w$ is a string

  - We construct $\mathcal{T}_M$ to decide $\mathcal{A}_{\text{TM}}$ as follows:

**Assume that $\mathcal{E}_{\text{LBA}}$ is decidable. Let $\mathcal{R}$ be a TM that decides it**
Continued: Can a CFG accept such a language?

• Let's think of this in terms of a PDA $D$ (with non-determinism)

- Can just process the input tape from left to right
- Non-deterministically guess where the error is
- Can just process the input tape from left to right

Let's think of this in terms of a PDA $D$ (with non-determinism)

Can we determine if a CFG accepts everything?

• Can use reduction via computation histories to establish:

Theorem: $\forall T \in \text{CFG}$ is undecidable

$\{\langle G, w \rangle \mid G \text{ is a CFG and } T(G(w)) = 1 \}$

undecidability of certain problems relative to CFG and PDA.
Proof

Assume that \( \text{ALL}_{\text{CFG}} \) is decidable. Let \( R \) be a TM that decides it.

1. Construct PDA \( D \) from \( M \) and \( w \) as described on previous slide.
   We can do this because we can algorithmically describe how to build \( D \).

2. Convert \( D \) to a CFG \( G \) (from Chapter 2).

3. Run \( R \) on input \( \langle G \rangle \).

4. If \( R \) rejects (not \( \Sigma^* \)), accept; if \( R \) accepts, reject.

Aside:

A CFG cannot determine if a computation history is valid.

- If \( \mathcal{C}_i \) is pushing \( \{ \mathcal{C}_{i+1} \}^+ \) onto the stack so it can compare it to \( \mathcal{C}_i \), it would need to validate \( \mathcal{C}_{i+1} \) after popping \( \mathcal{C}_i \) off the stack.
- So, as it is popping \( \mathcal{C}_i \) off the stack to compare it to \( \mathcal{C}_{i+1} \), it would need to validate \( \mathcal{C}_i \) to make sure that \( \mathcal{C}_{i+1} \) is valid given \( \mathcal{C}_i \) and that \( \mathcal{C}_i \) is a single correct or incorrect transition.

- A TM cannot determine if a computation history is valid.
Post Correspondence Problem

- Are there other problems, not concerned with Automata, that are undecidable?

- Post Correspondence Problem
  - Collection of dominos e.g. \{ [bca], [aab], [ca], [abc] ]
  - Is it possible to make a list of these dominos (repetitions allowed) with the same string on top as on bottom (called a match)?
  - Some collections of dominos do not have a match: e.g. \{ [abcab], [ca], [accba] \}
  - Problem is to determine whether a collection of dominos has a match.

More formally:

- An instance of the PCP is a collection of dominos \( P \) and a match is a sequence \( i_1, i_2, ..., i_l \), where \( t_{i_1}t_{i_2}...t_{i_l} = b_{i_1}b_{i_2}...b_{i_l} \).

- PCP = \{ \langle P \rangle | P \) is an instance of PCP with a match \}

Overview

- Reductions via Computational Histories

- Mapping Reducibility
  - Post Correspondence Problem \( \leq \) Post Correspondence Problem
  - Reductions via Computational Histories
Finite Number of Tiles to End

• Problem: Infinite number of tiles to specify end configurations
• All that is needed is to check that the state is an accepting state
  - For each accepting state $q_f$, create a tile with state on top $[q_f]$
  - For each letter $a \in \Sigma$, create a tile for letter on top $[a]$
  - But what prevents a match from using these special tiles before the last configuration?

We have just created a way to match that is not a legal computation history

But when we ensure a match from using these special tiles before the last configuration,

- For each letter $a \in \Sigma$, create a tile for letter on top $[a]$,
- For each accepting state $q_f$, create a tile with state on top $[q_f]$.

All that is needed is to check that the state is an accepting state.

Problem: Infinite number of tiles to specify end configurations.

Is PCP decidable?

• Show that ATM can be reduced to PCP
• Use dominoes to describe computation histories
• A match will be a successful computation of ATM on $w$
• A domino with $C_i$ on bottom, nothing on top (on bottom)
• A domino with $C_i$ on bottom, nothing on top (on top)
• For all of the legal ways that any $C_i$ (on top) can yield some $C_{i+1}$
• Matching makes sure $C_i$'s on bottom makes $C_i$'s on top

Simplistic View:

A match will be a successful computation of ATM on $w$

- Use dominoes to describe computation histories
- A match will be a successful computation of ATM on $w$

- $\vdash_{\Sigma^\star}$ can be reduced to PCP
- $\vdash_{\Sigma^\star}$ can be reduced to PCP

- What's wrong with our simplistic view?

$\vdash_{\Sigma^\star}$ can be reduced to PCP
Finite number of dominos for Transition

- Use dominos that we added earlier.
- Rest of transition from \( C_i \) to \( C_{i+1} \) stays the same.
- If \( \delta(1,0,0) = (0,1,\text{R}) \), use \( \begin{bmatrix} 1 \\ 0 \end{bmatrix} \).
- Just pull this part of configurations on dominos.
- Position of head (forward/backward).
- Next configuration \( C_{i+1} \) might change.
- Content to the left of head in \( C_i \).
- Same.
- Next configuration \( C_{i+1} \) depends on

A Better Way to End

- Allow accept state to swallow up a character before it or after it.
- For each \( a \in \Sigma \), add \( \begin{bmatrix} a \\ \text{accept} \end{bmatrix} \).
- Use \( \begin{bmatrix} \text{accept} \\ \# \# \end{bmatrix} \) to end.

This allows us to alter any configuration that has an accept state.

- Think of this as changing TM so that once it gets to accept state, it adds extra transitions to empty out the tape.
- You'll take a number of pseudo-steps to end.
- For each \( a \in \Sigma \), and \( \text{accept} \).
- Allow accept state to swallow up a character before it or after it.
Modified Post Correspondence Problem

- Let $\star, \diamond$ be two characters not used in the tiles
- Convert every domino so that top has $\star$ before every character and bottom has $\star$ after each character
- Add extra domino for start domino that also has $\star$ before bottom string
- Add a special domino to start is to use the special start domino
- Only way to start is to use the special start domino

Can we get PCP to start with a certain tile without forcing it?

- All intermediate solutions cannot reuse the special start tile
- Intermediate tiles' first character of the top and bottom strings won't be the same
- Only way to start is to use the special start domino

Can MPCP be reduced to PCP?

- Add extra domino for start domino that also has $\star$ before bottom string
- Convert every domino so that top has $\star$ before every character and bottom has $\star$ after each character
- Let $\star$ be two characters not used in the tiles
- Can we get PCP to start with a certain tile without forcing it?

Almost There

- Use Modified Post Correspondence Problem
  - Must start with a special domino
  - Acceptance can be reduced to MPCP
  - Acceptance problem is undecidable, so is $\text{MPCP}^p$
  - Given a TM $M$, a TM $S$ can build a MPCP that has a match iff $M$ accepts $w$
  - A TM can transform any $M$ into an equivalent one that does this by adding
  - A TM can transform any $M$ into an equivalent one that does this by adding
  - Assume $M$ is a TM that checks for the beginning of tape character
  - In an extra character to denote the beginning of the tape and extra
  - A TM can transform any $M$ into an equivalent one that does this by adding
  - Now can use domino that same letter on top and bottom to end right away
  - We can use it to force the domino with $\star$ on bottom to be used first
  - Only way to start is to use the special start domino
  - Can MPCP be reduced to PCP?
Overview

• Reductions via Computational Histories
  → Mapping Reducibility

Example of Converting MPCP to PCP

Tiles: \{ \[a\] \[b\] \[c\] \[d\] \}

Start tile: \[a\]

Tiles:

\[
\begin{array}{ccc}
  \[a\] & \[b\] & \[c\] \\
  \[d\] & \[e\] & \[f\] \\
\end{array}
\]

Example of Converting MPCP to PCP

Tiles: \{ \[a\] \[b\] \[c\] \[d\] \}

Start tile: \[a\]
Computable Function

Definition: A function $f : \Sigma^* \to \Sigma^*$ is a computable function if some Turing machine $M$, on every input $w$, halts with $f(w)$ on its tape.

- All arithmetic operations on integers are computable functions.
- Can make a machine that takes input $\langle m, n \rangle$ and computes sum of $m$ and $n$.
- Can make a machine that takes input $\langle w \rangle$ and reverses some of $w$ and $n$.
- Can make a machine that inserts a special character at the beginning of the tape when it starts (and moving all other characters one space to the right) and has special states if it reads that character later on.
- Inverse: A function $f : \Sigma^* \to \Sigma^*$ is a computable function if $f$ preserves acceptance (can be solved if $f$ accepts).
- If instance of $B$ is true (accept), so is instance of $A$.
- If we have such a reduction, we can solve $A$ with a solver for $B$.
- If there are instances of problem $B$ that covers instances of problem $A$ to $\Sigma^*$, then there is a computable function that converts instances of problem $A$ to instances of problem $B$.

Idea: Reducing problem $A$ to problem $B$ means that there is a more efficient way.

Let's formalize the notion of reducibility so that we can use it.

Used reducibility to prove a number of problems are undecidable.
Theorem: If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable.

Proof:
- Let \( M \) be a decider for \( B \) and \( f \) be the reduction from \( A \) to \( B \).
- A decider \( N \) for \( A \) is as follows:
  1. Compute \( f(w) \) on input \( w \).
  2. Run \( M \) on input \( f(w) \) and output whatever \( M \) outputs.
- Is \( N \) a decider of \( A \)?
  - Clearly if \( w \in A \), then \( f(w) \in B \) because \( f \) is a reduction from \( A \) to \( B \).
  - Thus, \( N \) accepts \( w \) whenever \( w \in A \). Therefore \( N \) works as desired.

Corollary: If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.

Definition: Language \( A \) is mapping reducible to language \( B \), written \( A \leq_m B \), if there is a computable function \( f : \Sigma^* \rightarrow \Sigma^* \) where for every \( w \), the following holds:

Formal Definition:

- A mapping reduction of \( A \) to \( B \) provides a way to convert questions about membership testing in \( A \) to membership questions in \( B \).
We use mapping reducibility. Let’s use mapping reducibility.

Let $\text{HALT}_{TM}$ be the halting problem. We want to make a computable function $f$ that takes input $\langle M, w \rangle$ and returns output $\langle M', w' \rangle$ where $\langle M, w \rangle \in \text{HALT}_{TM}$ if and only if $\langle M', w' \rangle \in \text{HALT}_{TM}$.

The following machine $F$ computes a reduction $f$:

1. Construct the following machine $M' = M$.
2. On input $x$:
   a. Run $M$ on $x$.
   b. If $M$ accepts, accept.
   c. If $M$ rejects, enter a loop.
3. Output $\langle M', w \rangle$.

Minor Issue: If $F$ determines its input is not in right format, just has to output some string not in $\text{HALT}_{TM}$.

Now let's use mapping reducibility. So our assumption is false.

- So is $\text{HALT}_{TM}$ decidable. Contradiction. So our assumption is false.
- If $f$ is computable, $\langle M', w \rangle$ exactly when $M$ accepts $w$. So it decides $\text{HALT}_{TM}$.
- $f$ always halts, and so is a decider.
- $f$ decides $\text{HALT}_{TM}$.

We construct $f$ as follows:

1. Run TM $H$ on input $\langle m, M, w \rangle$ where $\langle m, M, w \rangle \in \text{HALT}_{TM}$ and only if $\langle m, M, w \rangle \in \text{HALT}_{TM}$.
2. If $H$ accepts, accept.
3. If $H$ rejects, reject.
4. $\langle m, M, w \rangle \in \text{HALT}_{TM}$ if and only if $\langle m, M, w \rangle \in \text{HALT}_{TM}$.

Either way, $f$ is decidable.
Complementation

In previous proof

- Assume that $E_{TM}$ is decidable, say by $R$.

  + On input $\langle M, w \rangle$,
    
    + Construct $M_1$ that on input $x \neq w$ rejects, otherwise ran $M$ on $w$.

  + If $R$ accepts $M_1$, then $M_1$ accepts no strings, so $M_1$ rejects $w$, so accept.
  + Otherwise, reject.

  - Mapping reduction cannot change accepts and rejects.

Mapping reduction cannot change accepts and rejects

+ We do have a mapping reduction from $A_{TM}$ to $E_{TM}$.
  + Decidability is not affected by complementation, so $E_{TM}$ is undecidable.
+ We do not have a mapping reduction from $A_{TM}$ to $E_{TM}$.
  + In fact no such mapping reduction exists (Exercise 5.5).

More

- PCP is undecidable:
  - Contains two reductions $A_{TM} \leq_m M_{PCP}$ and then from $M_{PCP} \leq_m PCP$.
  - Mapping reducibility is transitive (Exercise 5.6).
  - So $A_{TM} \leq_m PCP$, and hence $A_{TM}$ is undecidable.

- Since $E_{TM}$ is undecidable, so is $\overline{E_{TM}}$.
  - So $\overline{A_{TM}} \leq_m PCP$, and hence $\overline{A_{TM}}$ is undecidable.
  - Since $\overline{A_{TM}}$ is undecidable, so is $\overline{PCP}$.

- Mapping reducibility is transitive (Exercise 5.6).
- Contains two reductions $\overline{A_{TM}} \leq_m \overline{M_{PCP}}$ and then from $\overline{M_{PCP}} \leq_m \overline{PCP}$.

- So $\overline{A_{TM}} \leq_m \overline{PCP}$, and hence $\overline{A_{TM}}$ is undecidable.

- Therefore, $PCP$ is undecidable.

- Therefore, $EQ_{TM}$ is undecidable.
Theorem: If $A \leq^m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.

Corollary: If $A \geq^m B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.