The Post Correspondence Problem

- Given a set, P of pairs of strings:

$$P = \{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}], \cdots, [\frac{t_k}{b_k}]\}$$

where $t_i, b_i \in \Sigma^*$

- **Question:** Does there exist a sequence $i_1, i_2, \cdots i_n$ such that:

$$t_{i_1} t_{i_2} \cdots t_{i_n} = b_{i_1} b_{i_2} \cdots b_{i_n}?$$

Note: the same pair can occur multiple times, i.e. there can be $j \neq m$ s.t. $i_j = i_m$.

A PCP Example

Example

Let P =

$$\left\{ \left[\frac{\mathtt{a}}{\mathtt{ab}}\right]_1, \left[\frac{\mathtt{ab}}{\mathtt{bb}}\right]_2, \left[\frac{\mathtt{ba}}{\mathtt{aa}}\right]_3, \left[\frac{\mathtt{bc}}{\mathtt{cc}}\right]_4, \left[\frac{\mathtt{ca}}{\mathtt{aa}}\right]_5, \left[\frac{\mathtt{cd}}{\mathtt{d}}\right]_6 \right\}$$

(I've numbered the tiles to make it easier to talk about them.)

Does the PCP problem P have a solution?

Another PCP Example

Example

Let P =

$$\left\{ \left[\frac{\mathtt{a}}{\mathtt{ab}}\right]_1, \left[\frac{\mathtt{b}}{\mathtt{cc}}\right]_2, \left[\frac{\mathtt{c}}{\mathtt{b}}\right]_3, \left[\frac{\mathtt{c}}{\mathtt{d}}\right]_4, \left[\frac{\mathtt{ddddd}}{\ }\right]_5, \left[\frac{\mathtt{ddde}}{\mathtt{e}}\right]_6 \right\}$$

(I've numbered the tiles to make it easier to talk about them.)

Does the PCP problem P have a solution?

- P has a solution iff $\exists n, (2^n \ mod \ 5) = 3$
- Yes, let $n = 3$.

PCP is undecidable

Theorem

The PCP problem is undecidable

Proof sketch

- Start with a pair that has the initial configuration for a TM on the bottom and an empty string on top.
- Include pairs in P whose top strings match the current configuration, and whose bottom strings build the next configuration.
- A bunch of details to:
  - Account for moving the tape head.
  - Extend the tape with blanks when needed.
  - Force the first pair of a solution to be the one that gives the initial configuration
  - ...

A Simplifying Assumption: We'll assume that any solution must start with tile 1 – we'll call this the **Modified Post Correspondence Problem** (MPCP). (Dont worry.) We'll remove this assumption later.

Proof – Tile 1

- We'll reduce $L_u = \{M\#w \mid M \text{ accepts } w\}$ to MPCP.
- Let $M\#w$ be a string where M describes a TM and w describes an input string to M.
- The first tile will give the initial TM configuration as the bottom string, and an empty string on top. Wefll use $\#$ (with $\# \notin \Gamma$) as the end marker for configurations.

$$\begin{array}{|c|}\hline \# \\ \hline \#q_0w\# \\ \hline \end{array}_{\,1} \quad \in \quad P$$

From one configuration to the next

- At each step, we copy the current configuration from the bottom string to the upper string, and build the next configuration on the lower string:

$$\begin{array}{|l|}\hline \#C_0\#C_1\#\ldots C_{k-1}\# \\ \hline \#C_0\#C_1\#\ldots C_{k-1}\#C_k \\ \hline \end{array} \rightarrow \begin{array}{|l|}\hline \#C_0\#C_1\#\ldots C_{k-1}\#C_k\# \\ \hline \#C_0\#C_1\#\ldots C_{k-1}\#C_k\#C_{k+1} \\ \hline \end{array}$$

- A configuration looks like $\alpha bqc\beta$.
- To calculate the next configuration, we
  - Copy α to the upper and lower strings.
  - Copy αbqc to the upper string and write its successor to the lower string.
  - Copy β to the upper and lower strings.
- To copy α and β we include the following tile in P for each $c \in \Gamma$: $\left[\frac{c}{c}\right]$
- The next two slides describe how to handle transitions.

All the Right Moves

For each transition $\delta(q, c) = (q', c', R)$:

- We add the tile $[\frac{qc}{c'q'}]$ to P. This enables the move:

$$\begin{array}{|l|}\hline \#\ldots\#\alpha \\ \hline \#\ldots\#\alpha qc\beta\#\alpha \\ \hline\end{array} \rightarrow \begin{array}{|l|}\hline \#\ldots\#\alpha qc \\ \hline \#\ldots\#\alpha qc\beta\#\alpha c'q' \\ \hline\end{array}$$

- If $c =$ blank symbol, we also add the tile $[\frac{q\#}{c'q'\#}]$ to handle the case when the head is moving further into the infinite string of blanks at the end of the tape.

All the Left Moves

For each transition $\delta(q, c) = (q', c', L)$:

- for each $b \in \Gamma$ we add the tile $[\frac{qbc}{q'bc'}]$ to P. This enables the move:

$$\begin{array}{|l|}\hline \# \ldots \# \alpha \\ \hline \# \ldots \# \alpha b q c \beta \# \alpha \\ \hline \end{array} \rightarrow \begin{array}{|l|}\hline \# \ldots \# \alpha b q c \\ \hline \# \ldots \# \alpha b q c \beta \# \alpha q' b c' \\ \hline \end{array}$$

- We also add the tile $[\frac{\# qc}{\# q'c'}]$ to P to handle the case when the head is at the left end of the tape.

The End Game

- M accepts w iff we can reach a configuration for our MPCP

$$\left[\frac{\#C_0 \ldots \#C_{n-1}\#}{\#C_0 \ldots \#C_{n-1}\#\alpha q_{accept}\beta\#}\right]$$

- Now we have to fix the problem that we've got one more configuration on the lower tape than the upper one. For each $c \in \Gamma$ we add the tiles: $\left[\frac{c q_{accept}}{q_{accept}}\right], \left[\frac{q_{accept} c}{q_{accept}}\right].$
- These allow us to discard one tape symbol each time we copy the configurations until we get to: $\left[\frac{\#C_0 \ldots \#q_{accept}c\#}{\#C_0 \ldots \#q_{accept}c\#q_{accept}\#}\right]$

So, we add one more tile to our set: $\left[\frac{q_{accept}\#\#}{\#}\right].$

From MPCP to PCP

- We need to force our tile1 to be the first tile of any solution.
- Let $\star$ be a new symbol (i.e. not in $\Gamma \cup \{\#\}$).
- For any string s, let $\star s$ be the string obtained by inserting a $\star$ before each symbol of s. For example, $\star(abc) = \star a \star b \star c$.
- For any string s, let $s\star$ be the string obtained by adding a $\star$ before each symbol of s. For example, $(abc)\star = a \star b \star c\star$.
- Finally, $\star s\star$ puts on star between each pair of symbols of s and one star at the beginning of s and one at the end. For example, $\star(abc)\star = \star a \star b \star c\star$.

From MPCP to PCP

- Given a set of tiles, P for MPCP as described above:

  Replace the initial tile, $\left[\frac{\#}{\#q_0w\#\star}\right]$ with $\left[\frac{\star\#}{\star\#q_0w\#\star}\right]$.

  Replace the final tile, $\left[\frac{q_{accept}\#\#}{\#}\right]$ with $\left[\frac{\star q_{accept}\#\star\#}{\#}\right]$ with

  For every other tile, $\left[\frac{t}{b}\right]$, replace it with $\left[\frac{\star t}{b\star}\right]$

- Now $\left[\frac{\star\#}{\star\#q_0w\#\star}\right]$ must be the first tile of any solution because it is the only tile that starts and ends with the same symbol.
- We have reduced computational histories for L_u to PCP. Hence, PCP is undecidable.

The CFG Ambiguity Problem

- Use the lists $[\frac{1}{10}]_a, [\frac{0}{10}]_b, [\frac{010}{01}]_c, [\frac{11}{1}]_d$

  The grammar is
  - $S \to A \mid B$
  - $A \to 1Aa \mid 0Ab \mid 010Ac \mid 11Ad \mid \epsilon$
  - $B \to 10Ba \mid 10Bb \mid 01Bc \mid 1Bd \mid \epsilon$

Each string has a unique derivation from A and B
Ambiguity can only come from S.

Theorem

It is undecidable to determine whether a given CFG G is ambiguous.

Is the Intersection of Two CFL's Empty?

Consider the two list languages from a PCP instance. They have an empty intersection if and only if the PCP instance has no solution.

Theorem

Given two CFLs L_1 and L_2, "$L_1 \cap L_2 = \emptyset$?" is undecidable

Complements of List Languages

- We can get other undecidability results about CFL's if we first establish that the complement of a list language is a CFL.
- PDA is easier approach.
- Accept all ill-formed input (not a sequence of symbols followed by indexes) using the state.
- For inputs that begin with symbols from the alphabet of the PCP instance, store them on the stack, accepting as we go.
- When index symbols start, pop the stack, making sure that the right strings were found on top of the stack; again, keep accepting untilK
- When we expose the bottom-of-stack marker, we have found a sequence of strings from the PCP list and their matching indexes. This string is not in the complement of the list language, so don't accept.
- If more index symbols come in, then we have a mismatch, so start accepting again and keep on accepting.

Is a CFL Equal to Σ^*?

Take an instance of PCP, say lists A and B. The union of the complements of their two list languages is Σ^* if the instance has no solution, and something less if there is a solution.

Theorem

Given a CFL L, "$L = \Sigma^*$?" is undecidable

Unrestricted Grammars

- A grammar (V, T, S, P) is *unrestricted* if all the productions are of the form $u \to v$, where u is in $(V \cup T)^+$ and v is in $(V \cup T)^*$.
  - Basically, no restrictions imposed on productions
  - Any number of variables on the left and right-hand sides
  - Only restriction is that ϵ cannot appear on the left side of a production
- Any language generated by an unrestricted grammar is recursively enumerable

Context-Sensitive Grammars

- Between the unrestricted grammars and the "restricted" CFGs, there is a variety of "somewhat restricted" grammars
- A grammar is context-sensitive if all productions are of the form $x \to y$, where x, y are in $(V \cup T)^+$ and $|x| \leq |y|$
  - Fundamental property:
    - ★ grammar is non-contracting– i.e., the length of successive sentential forms can never decrease
  - Why "context-sensitive"?
    - ★ All productions can be rewritten in a normal form $xAy \to xvy$
    - ★ Effectively, "A can be replaced by v only in the context of a preceding x and a following y"

An Example

CSG for $\{a^n b^n c^n | n \geq 1\}$

$$S \to abc | aAbc$$
$$Ab \to bA$$
$$Ac \to Bbcc$$
$$bB \to Bb$$
$$aB \to aa | aaA$$

$S \Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \Rightarrow aBbbcc \Rightarrow aaAbbcc \Rightarrow aabAbcc \Rightarrow aabbAcc \Rightarrow aabbBbccc \Rightarrow aabBbbccc \Rightarrow aaBbbbccc \Rightarrow aaabbbccc$

A and B are "messengers"- an A is created on the left, travels to the right to the first c, creates another b and c. Then sends B back to create the corresponding a. Similar to the way one would program a TM to accept the language.

Linear-Bounded Automata

- A limited TM in which tape use is restricted
  - Use only part of the tape occupied by the input
  - I.e., has an unbounded tape, but the mount that can be used is a function of the input
    - ★ Restrict usable part of tape to exactly the cells taken by the input

- LBA is assumed to be nondeterministic

Theorem

L is accepted by some linear bounded automaton iff there is a context-sensitive grammar that generates L.

The Expanded Chomsky Hierarchy