Post's Correspondence Problem

- An undecidable, but RE, problem that appears not to have anything to do with TM's.
- Given two lists of "corresponding" strings $(w_1, w_2, \ldots, w_n)$ and $x_1, x_2, \ldots, x_n)$, does there exist a nonempty sequence of integers $i_1, i_2, \ldots, i_k$ such that $w_{i_1} w_{i_2} \cdots w_{i+k} = x_{i_1} x_{i_2} \cdots x_{i_k}$?
- Intuition: we can try all lists $i_1, i_2, \ldots, i_k$ in order of k. If we find a solution, the answer is "yes." But if we never find a solution, how can we be sure no longer solution exists, so we can never say "no."

Example

$(1, 0, 010, 11)$ and $(10, 10, 01, 1)$.

- A solution is $1, 2, 1, 3, 3, 4$.
- The constructed string from both lists is 10101001011.

Another Example

From the course reader: $(10, 011, 101)$ and $(101, 11, 011)$.

- Another argument why this instance of PCP has no solution:
  - The first index has to be 1 because only the pair 10 and 101 begin with the same symbol.
  - Then, whatever indexes we choose to continue, there will be more 1's in the string constructed from the first list than the second (because in each corresponding pair there are at least as many 1's in the second list). Thus, the two strings cannot be equal.

Plan to Show PCP is Undecidable

1. Introduce MPCP, where the first pair must be taken first in a solution.
2. Show how to reduce MPCP to PCP.
3. Show how to reduce L_u to MPCP.
   - This is the only reason for MPCP: it makes the reduction from L_u easier.
4. Conclude that if PCP is decidable, so is MPCP, and so is L_u (which we know is false).

Reduction of MPCP to PCP

Trick: given a MPCP instance, introduce a new symbol *.

- In the first list, * appears after every symbol, but in the second list, the * appears *before* every symbol.
  - Example: the pair 10 and 011 becomes 1*0* and *0*1*1.
  - Notice that no such pair can ever be the first in a solution.
- Take the first pair w_1 and x_1 from the MPCP instance (which must be chosen first in a MPCP solution) and add to the PCP instance another pair in which the *'s are as always, but w_1 also gets an extra * at the beginning.
  - Referred to as "pair 0."
  - Example: if 10 and 011 is the first pair, also add to the PCP instance the pair *1*0* and *0*1*1.
- Finally, since the strings from the first list will have an extra * at the end, add to the PCP instance the pair $ and *$.
  - $ is a new symbol, so this pair can be used only to complete a match.
  - Referred to as the "final pair."

Proof the Reduction is Correct

- If the MPCP instance has a solution 1 followed by $i_1, i_2, \ldots, i_k$, Then the PCP instance has a solution, which is the same, using the first pair in place of pair 1, and terminating the list with the final pair.
- If the PCP instance has a solution, then it must begin with the "first pair," because no other pair begins with the same symbol. Thus, removing the *'s and deleting the last pair gives a solution to the MPCP instance.

Reduction of L_u to MPCP

- Intuition: The equal strings represent a computation of a TM M on input w.
  - Sequence of ID's separated by a special marker #.
- First pair is # and $\#q_0w\#$.

- String from first list is always one ID behind, unless an accepting state is reached, in which case the first string can "catch up."
- Some example pairs:

1. X and X for every tape symbol X. Allows copying of symbols that don't change from one ID to the next.
2. If $\delta(q, X) = (P, Y, R)$, then qX and Yp is a pair. Simulates a move for the next ID.
3. If q is an accepting state, then XqY and q is a pair for all X and Y. Allows ID to "shrink to nothing" when an accepting state is reached.

Undecidable Problems About CFL's

We are applying the theory of undecidability in a useful way when we show a real problem not to be solvable by computer.

- Example: You may think the CS154 project was hard, but at least there is an algorithm to convert RE's to DFA's, so it is at least possible for you to succeed.
- Suppose next spring's CS154 project is to take a CFG and tell whether it is ambiguous. You can't do it because the problem is undecidable!

Converting PCP to CFG's

For each list $A = (w_1, w_2, \ldots, w_n)$ we can construct a grammar and a language that represents all sequences of integers $i_1, i_2, \ldots, i_k$ and the strings $w_{i_1} w_{i_2} \cdots w_{i_k}$ that are constructed from those lists of integers.

- Use $a_1, a_2, \ldots, a_n$ as new symbols (not in the alphabet of the PCP instance) representing the integers.
- The "grammar for list A": $S \rightarrow w_1 S a_1 \mid w_2 S a_2 \mid \cdots \mid w_n S a_n \mid \epsilon$.
    - ❖ Yields all concatenations of w's followed by the reverse of their index sequence.

Reduction of PCP to CFG Ambiguity Problem

Given lists A and B, construct grammar as follows:

- $S \rightarrow A \mid B$.
- A is the start symbol for a grammar from list A; B is the same for list B.

- If there is a solution to the PCP instance, then the same string can be derived starting $S \underset{lm}{\Rightarrow} A$ and $S \underset{lm}{\Rightarrow} B$.
    - Conversely, the only way a string can have two leftmost derivations is if they begin in these two ways, because the grammar of one list is unambiguous.

Example

Use the lists of our first example: $(1, 0, 010, 11)$ and $(10, 10, 01, 1)$. Let a, b, c, d stand for the four index integers. The grammar is:

$$S \rightarrow A \mid B$$
$$A \rightarrow 1Aa \mid 0Ab \mid 010Ac \mid 11Ad \mid \epsilon$$
$$B \rightarrow 10Ba \mid 10Bb \mid 01Bc \mid 1Bd \mid \epsilon$$

- A string with two leftmost derivations: $10101001011dccaba$.
    - $S \underset{lm}{\Rightarrow} 1Aa \underset{lm}{\Rightarrow} 10Aba \underset{lm}{\Rightarrow} 101Aaba \underset{lm}{\Rightarrow} 101010Acaba \underset{lm}{\Rightarrow} 101010010Accaba \underset{lm}{\Rightarrow} 10101001011Adccaba \underset{lm}{\Rightarrow} 10101001011dccaba$.
    - $S \underset{lm}{\Rightarrow} 10Ba \underset{lm}{\Rightarrow} 1010Bba \underset{lm}{\Rightarrow} 101010Baba \underset{lm}{\Rightarrow} 10101001Bcaba \underset{lm}{\Rightarrow} 1010100101Bccaba \underset{lm}{\Rightarrow} 10101001011Bdccaba \underset{lm}{\Rightarrow} 10101001011dccaba$.

Undecidable Problem: 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 a solution.

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 until$\cdots$
- 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.

Undecidable Problem: 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.

Undecidable Problem: Is the Intersection of Two CFL's Regular?

Key idea: the intersection of list languages is regular if and only if it is empty. Thus, PCP reduces to regularity of intersection for CFL's.

- Obviously, if empty, it is regular.
- Suppose the intersection of two list languages, for A and B, $L_A \cap L_B$, is nonempty. Then there is a solution to this instance of PCP, say string w and string of index symbols i.
  - ❖ Example: for the running PCP instance, $w = 10101001011$ and $i = abaccd$.
- Then i^k is an index sequence that yields solution w^k for all k.
  - ❖ General principle: concatenation of PCP solutions is a solution.
- Consider homomorphism $h(0) = w$ and $h(1) = i^R$.
- $h^{-1}(L_A \cap L_B)$ is $\{0^n 1^n \mid n \geq 1\}$.
- Since regular languages are closed under inverse homomorphism, if the intersection were regular, so would $h^{-1}(L_A \cap L_B)$ be.
- Since we know this language is not regular, we conclude that $L_A \cap L_B$ is not regular.