Computability

Post’s Correspondence Problem
Valid Turing Computations and other results

Instance of PCP

- Take two lists of strings over the same alphabet $\Sigma$: $A = w_1, w_2, \ldots, w_k$ and $B = x_1, x_2, \ldots, x_k$ for the same $k$
- PCP has a solution if we can find $i_1 w_{i_2} w_{i_3} \cdots w_i m = x_{i_1} x_{i_2} x_{i_3} \cdots x_{i_m}$
- PCP is undecidable

The CFG ambiguity problem

- Take any PCP over alphabet $\Sigma$: $A = w_1, w_2, \ldots, w_n$ and $B = x_1, x_2, \ldots, x_n$ for some $n$
- Let $a_1, a_2, \ldots, a_n$ be $n$ new symbols
- Define $L_A = \{w_{i_1} w_{i_2} \cdots w_{i_m} a_{i_m} a_{i_m-1} \cdots a_{i_1} \mid m \geq 1\}$ and $L_B = \{x_{i_1} x_{i_2} \cdots x_{i_m} a_{i_m} a_{i_m-1} \cdots a_{i_1} \mid m \geq 1\}$
- CFL’s and ambiguity is equivalent to solving PCP
Valid computations

Section 8.6: valid computations of TM’s

- For use in section 8.6, a valid computation of a Turing machine \( M = (Q, \Sigma, \Gamma, \delta, q_0, B, F) \) is a string
  \( w_1\#w_2\#^Rw_3\#^Rw_4\#^R\ldots \), where each \( w_i \) is an i.d. in a sequence of i.d.’s of the
  TM which accepts \( w \), where \( w_1 = q_0w \)
- The sequence ends with \( w_n\# \) or \( w_n\#^R \) where \( w_n \) contains a state in \( F \)
- Each \( w_i \) is a string in \( \Gamma^*Q\Gamma^* \) that does not end in a \( B \)

Two results on valid computations

- The textbook shows (Lemma 8.6 and Lemma 8.7) that the valid
  computations of a TM form a
  language which is the intersection of
  2 CFL’s and the complement of that
  language is also a CFL
- In fact, given a TM \( M \), the textbook
  shows how to construct grammars
  for the languages needed
- First we describe \( L_1 \cap L_2 \)

Lemma 8.6

- Lemma 8.6 shows that there is a PDA
  for \( L_3 = \{ y \# z^R : y \vdash_M z \} \)
- A PDA can pass the elements of
  \( y = \alpha q \beta \) onto a stack but making the
  transition of \( M \) as it encounters \( q \) and
  the following symbol
- Then the PDA checks \( z^R \) exactly
  matches what is on the stack and
  accepts by empty stack
The CFL’s

- \( L_1 = (L_3 \#)^*(\{\varepsilon\} \cup \Gamma^* F \Gamma^* \#) \)
  - to allow for the last ID of \( w_1 \# w_2 R \# w_3 \# w_4 R \# \ldots \)
  - to be a \( w_n \) or a \( w_n^R \)
- You then make a PDA for \( L_4 = \{ yR \# z : y \vdash_M z \} \) and define
- \( L_2 = \delta \Sigma^* \# (L_4 \#)^*(\{\varepsilon\} \cup \Gamma^* F \Gamma^* \#) \)
- One or other of \( L_1 \) and \( L_2 \) contains the accepting ID of the TM (or its reverse)

---

Theorem 8.10

- \( L(G_1) \cap L(G_2) = \emptyset \) is undecidable
- Proof: suppose that it was decidable that given any two CFG’s then the intersection of their grammars is or is not empty, then \( L_e \) is decidable:
  - Then given any TM \( M \), we can construct the CFG’s of the CFL’s such that \( L_1 \cap L_2 \) is the set of valid computations of \( M \)
  - \( L(M) = \emptyset \) if and only if \( L_1 \cap L_2 = \emptyset \)

---

Lemma 8.7

- An invalid computation of a Turing Machine is a string in \( (\Gamma \cup Q \cup \{\#\})^* \)
  - that is not of the form of a valid computation
  \( w_1 \# w_2 R \# w_3 \# w_4 R \# \ldots \)
  - (where the \( w \)’s satisfy all the conditions of a valid computation)
- Lemma 8.7 states that the set of invalid computations is a CFL
- It is a union of 3 regular sets and 2 CFL’s

Outline Proof

- What is not a valid computation?
- \( \{x_1 \# x_2 \# x_3 \# x_4 \# \ldots x_n \# : \text{each } x_i \text{ is an ID of } M \text{ or the reverse of one} \}
  - is a regular language with regular expression: \( (\Gamma^* Q \Gamma^* \#)^* \) (the reverse of an ID is an ID)
- So the complement is also regular
- That is a large part of the invalid computations
An invalid computation is also one that does not begin with an element of $q_0\Sigma^*$.

Now $q_0\Sigma^*(\Gamma^*Q\Gamma^*)^* = \{x_1 \# x_2 \# x_3 \# x_4 \# \ldots x_n \# : \text{each } x_i \text{ is an ID of } M \text{ and } x_1 \text{ is an initial ID}\}$ is regular so its complement in $(\Gamma \cup Q \cup \{\#\})^*$ is regular.

An invalid computation is also one that does not end with a final state.

Now $(\Gamma^*Q\Gamma^*)^*(\Gamma^*F\Gamma^*)^* = \{x_1 \# x_2 \# x_3 \# x_4 \# \ldots x_n \# : \text{each } x_i \text{ is an ID of } M \text{ and } x_n \text{ is a final ID}\}$ is regular so its complement in $(\Gamma \cup Q \cup \{\#\})^*$ is regular.

What remains as invalid is a part of the following larger set of invalid computations:

$\{x_1 \# x_2 \# x_3 \# x_4 \# \ldots x_n \# : \text{each } x_i \text{ is an ID of } M \text{ and for some odd } i, x_i \vdash (x_{i+1})^R \text{ is false or for some even } i, x_i^R \vdash x_{i+1} \text{ is false}\}$

This set is the union of two CFL’s.

One of the two CFL’s is:

$\{x_1 \# x_2 \# x_3 \# x_4 \# \ldots x_n \# : \text{each } x_i \text{ is an ID of } M \text{ and for some odd } i, x_i \vdash (x_{i+1})^R \text{ is false}\}$

A PDA randomly starts analyzing after passing and even number of #’s.

As it reads $x_i$ it creates the correct next ID on the stack (as in Lemma 8.6).

Then it compares $x_{i+1}$ with the stack contents: if there is a mismatch the PDA moves to the end & succeeds.
The other set is 
{\(x_1 \# x_2 \# x_3 \# x_4 \# ... x_n \#\) : each \(x_i\) is an ID of \(M\) and for some even \(i\), \(x_i^R \vdash x_{i+1}\) is false}

This is a CFL for similar reasons

Hence the set of invalid computations is a CFL

Suppose it were decidable whether \(L(G) = \Sigma^*\), we could deduce that \(L(M) = \emptyset\) is decidable (which is false)

Take \(M\), and construct the grammar \(G\) of all invalid computations of \(M\)

Theorem 8.11

- \(L(M) = \emptyset\) if and only if
- all computations are invalid if and only if
- \(L(G) = \Sigma^*\)
- If there were an algorithm to test \(L(G) = \Sigma^*\), we could now check if \(L(M) = \emptyset\)

Consequences of Theorem 8.11

- Theorem 8.12
  Let \(G_1\) and \(G_2\) be arbitrary CFG’s and \(R\) a regular set. The following are undecidable:
  - \(L(G_1) = L(G_2)\),
  - \(L(G_2) \subseteq L(G_1)\),
  - \(L(G_1) = R\),
  - \(R \subseteq L(G_1)\)
Proof

- Pick $G_2$ so that $L(G_2) = \Sigma^*$ where $\Sigma$ is the alphabet of $G_1$, then (1) and (2) reduce to $L(G_1) = \Sigma^*$
- Pick $R = \Sigma^*$ then statements (3) and (4) reduce to determining if $L(G_1) = \Sigma^*$
- Deciding if $L(G_1) = \Sigma^*$ is undecidable (so the 4 problems are undecidable even in this special case)

Something that IS decidable

- For an arbitrary CFG $G$ and a regular language $R$, it is decidable that $L(G) \subseteq R$
- This property is equivalent to $L(G) \cap R = \emptyset$
- Since that intersection is a CFL, we can test if it is empty by checking if the start symbol is useful

Lemma 8.8

- Let $M$ be a Turing machine that makes at least 3 moves on every input, then the set of valid computations of $M$ is a CFL if and only if $L(M)$ is finite
- Proof:
  1. If $L(M)$ is finite, then the set of valid computations is finite, which is certainly a CFL
  2. (ii) if the set of valid computations were a CFL and $L(M)$ were infinite, then pick a valid computation starting $w_1 \# w_2^R \# w_3 \# ...$ where $w_1$ is long enough to make $w_2$ longer than the $n$ of Ogden’s lemma (an extension of PL4CFL)
  3. That allows to pump up on $w_2^R$ only and the computation becomes invalid
  4. Hence the valid comp’s are not a CFL
Theorem 8.13

- It is undecidable for arbitrary CFG’s $G_1$ and $G_2$ whether
  (i) $L(G_1)$ is a CFL
  (ii) $L(G_1) \cap L(G_2)$ is a CFL
- Proof. (i) Take any $M$ and convert it so that it makes 2 moves at least on all inputs
- Construct the grammar $G$ for all invalid computations
- $L(G)$ is a CFL iff $M$ accepts a finite set

But finiteness is undecidable
- Proof (ii) Take any $M$ and convert it to always make at least 2 moves
- Construct the $G_1$ and $G_2$ so that $L(G_1) \cap L(G_2)$ is the set of valid comps of $M$
- $L(G_1) \cap L(G_2)$ is a CFL iff $M$ accepts a finite set but
- Finiteness is undecidable

Section 8.7 - Greibach

- The textbook goes on to describe a result of Greibach and shows how to deduce the undecidability of the questions:
- whether a CFG generates a regular language
- inherent ambiguity of a CFL is undecidable

Greibach’s Theorem

- Let $C$ be a class of languages that is \textit{effectively} closed under concatenation with regular sets and union and for which “= $\Sigma^*$” is undecidable for any large $\Sigma$.
- Let $P$ be any non-trivial property that is true for all regular sets and that is preserved under $L \rightarrow \{w : wa \in L\}$
- Then $P$ is undecidable for $C$