CFL's and Noncomputability

These brief notes are intended to supplement the text **Introduction to the Theory of Computation** by Michael Sipser, Third Edition.

We are especially interested in the proof of Theorem 5.13 (ALL_{CFG} is undecidable). (See Sections 2.1, 2.2, and 2.4. for background on CFLs.)

Here we assume Theorem 2.20 (A language is context free iff some PDA (pushdown automaton) recognizes it). Thus (letting CFL denote the set of context free languages)

CFL $= \{\mathcal{L}(M) : M$ is a PDA$\}$.

From Section 2.4 we have the following definition:

DCFL $= \{\mathcal{L}(M) : M$ is a DPDA$\}$, where DCFL is the set of deterministic context free languages and DPDA stands for deterministic pushdown automaton.

Here are some facts about DCFL:

1) DCFL is closed under complementation.
2) DCFL is not closed under union, and not closed under intersection.
3) Both CFL and DCFL are closed under intersection with regular sets.

Here are proof sketches for the above:

1) If the $L = \mathcal{L}(M)$, where M is a DPDA, then $\overline{L} = \mathcal{L}(M')$, where M' is obtained from M by changing accept states to reject states and vice versa.
2) Define

$$\begin{aligned} L_{abc} &= \{a^n b^n c^n : n \geq 0\} \\ A_1 &= \{a^i b^j c^k : i, j, k \geq 0\} \\ A_2 &= \{a^i b^i c^j : i, j \geq 0\} \\ A_3 &= \{a^i b^j c^j : i, j \geq 0\} \end{aligned}$$

   Note that L_{abc} is NOT a context free language (this can be shown by the Pumping Lemma).

However it is easy to see that each of A_1, A_2 and A_3 is a deterministic CFL (and in fact A_1 is a regular language). Hence $\overline{A_1}$ is a regular language, and $\overline{A_2}$ and $\overline{A_3}$ are in DCFL.

It is not hard to see that

$$\overline{L_{abc}} = \overline{A_1} \cup \overline{A_2} \cup \overline{A_3} \tag{1}$$

(The union of the three sets represents the three reasons that a string might not be in L_{abc}.)

Since DCFL is closed under complementation and CFL is closed under union, it follows that $\overline{L_{abc}}$ is a context-free language. However $\overline{L_{abc}}$ is not a deterministic context free language, because DCFL is closed under complementation. It follows that DCFL is not closed under union. But then DCFL is not closed under intersection, since otherwise by De Morgan's laws, it would be closed under union.

3) Suppose that $L_1 = L_2 \cap L_3$, where L_2 is a regular set and L_3 is a context free language. Then L_2 is accepted by a FA M_2 and L_3 is accepted by a PDA M_3. We can design a PDA M_1 which accepts L_1 by simultaneously simulating M_2 and M_3, and M_1 accepts its input iff both M_2 and M_3 accept. (The states of M_1 consist of all pairs (q, q') where q is a state of M_2 and q' is a state of M_3.) If M_3 is deterministic then M_1 is also deterministic.

Now in order to show ALL_{CFG} is not semidecidable we show $\overline{\mathrm{HB}} \leq_m \mathrm{ALL}_{CFG}$. Given a Turing machine M we want to construct a CFG G such that M does not halt on a blank tape iff $\mathcal{L}(G) = \Sigma^*$.

We will construct G so that $\mathcal{L}(G)$ consists of all strings which do *not* code a halting of M computation starting with a blank tape. This suffices, because if M halts on a blank tape then $\mathcal{L}(G)$ is missing exactly one string, namely the string coding the halting computation of M on a blank tape. But if M does not halt on a blank tape, then $\mathcal{L}(G) = \Sigma^*$, as required.

We code computations of M by the sequence of configurations $C_1, C_2, \ldots$, except that the strings representing every second configuration are reversed (see Figure 5.14, page 226 in the text). The configurations are separated by the symbol #.

To construct the CFG G we use the idea from equation (1) above, except now L_{abc} is replaced by the language L_{comp}, which consists of all strings encoding a halting computation of M on a blank tape. Thus L_{comp} is either empty (if M does not halt) or consists of exactly one string.

Now we construct three languages L_1, L_2, L_3 (to correspond to A_1, A_2, A_3 in (1)), so

$$\mathcal{L}(G) = \overline{L_{comp}} = \overline{L_1} \cup \overline{L_2} \cup \overline{L_3}$$

where $\overline{L_1}, \overline{L_2}, \overline{L_3}$ represent the three reasons that a string might not code a halting computation of M on a blank tape.

Further $\overline{L_1}$ is a regular set, and $\overline{L_2}$ and $\overline{L_3}$ are deterministic CFLs.

Thus

1) L_1 is the set of all strings which begin with $\#q_0b\#$ and end with $\#u\#$ where u codes a halting configuration or its reverse, and the segment between any consecutive pair of $\#$'s codes a configuration (or its reverse).

2) L_2 is the set of all strings w such that for every segment of w of the form $\#u\#v\#$ (with the first $\#$ preceded by an *even* number of $\#$'s), if u codes a configuration C of M and v has no occurrence of $\#$, then v codes the reverse of the successor to the configuration C.

3) L_3 is the set of all strings w such that for every segment of w of the form $\#u\#v\#$ (with the first $\#$ preceded by an *odd* number of $\#$'s), if u codes the reverse of a configuration C of M and v has no occurrence of $\#$, then v codes the successor to the configuration C.

It is not hard to see that L_1 is accepted by some Finite Automaton, and both L_2 and L_3 are accepted by deterministic PDAs. Thus L_1 and $\overline{L_1}$ are regular sets, and $L_2, \overline{L_2}, L_3, \overline{L_3}$ are all deterministic CFLs.