CS 360 - MODULE 7 - ADDITIONAL NOTES

COLLIN ROBERTS

1. Example of Creating a CNF Grammar From An Arbitrary Grammar

Recall the grammar for palindromes over $\Sigma = \{0, 1\}$:

$$G : S \to \varepsilon|0|1|0S0|1S1.$$

Here we create a new CNF grammar G' such that $L(G) = L(G') \cup \{\varepsilon\}$.

(1) Nullable Variables The only variable, S, is nullable. We add the rules

$$S \to 00|11,$$

then we delete the rule $S \to \varepsilon$. Our modified grammar so far is

$$G_1 : S \to 0|1|00|11|0S0|1S1.$$

(2) Unit Productions There are no unit productions, so there is nothing to do at this step.

(3) Productions With > 2 Symbols on RHS

(a) $S \to 0S0$: Add X_1, and add rules

$$\begin{aligned} S &\to 0X_1 \\ X_1 &\to S0, \end{aligned}$$

then delete the production $S \to 0S0$.

(b) $S \to 1S1$: Add X_2, and add rules

$$\begin{aligned} S &\to 1X_2 \\ X_2 &\to S1, \end{aligned}$$

then delete the production $S \to 1S1$.

Our modified grammar so far is

$$\begin{aligned} G_2 : S &\to 0|1|00|11|0X_1|1X_2 \\ X_1 &\to S0 \\ X_2 &\to S1. \end{aligned}$$

(4) Productions With 2 Symbols on RHS Not Both Variables

(a) $S \to 00$: Add X_3, X_4, and add rules

$$\begin{aligned} S &\to X_3X_4 \\ X_3 &\to 0 \\ X_4 &\to 0, \end{aligned}$$

then delete the production $S \to 00$.

(b) $S \to 11$: Add X_5, X_6, and add rules

$$\begin{aligned} S &\to X_5X_6 \\ X_5 &\to 1 \\ X_6 &\to 1, \end{aligned}$$

then delete the production $S \to 11$.

(c) $\underline{S \to 0X_1}$: Add X_7, and add rules

$$\begin{aligned} S &\to X_7X_1 \\ X_7 &\to 0, \end{aligned}$$

then delete the production $S \to 0X_1$.

(d) $\underline{S \to 1X_2}$: Add X_8, and add rules

$$\begin{aligned} S &\to X_8X_2 \\ X_8 &\to 1, \end{aligned}$$

then delete the production $S \to 1X_2$.

(e) $\underline{X_1 \to S0}$: Add X_9, and add rules

$$\begin{aligned} X_1 &\to SX_9 \\ X_9 &\to 0, \end{aligned}$$

then delete the production $X_1 \to S0$.

(f) $\underline{X_2 \to S1}$: Add X_{10}, and add rules

$$\begin{aligned} X_2 &\to SX_{10} \\ X_{10} &\to 1, \end{aligned}$$

then delete the production $X_2 \to S1$.

The final grammar G' is therefore

$$\begin{aligned} S &\to 0|1|X_3X_4|X_5X_6|X_7X_1|X_8X_2 \\ X_1 &\to SX_9 \\ X_2 &\to SX_{10} \\ X_3 &\to 0 \\ X_4 &\to 0 \\ X_5 &\to 1 \\ X_6 &\to 1 \\ X_7 &\to 0 \\ X_8 &\to 1 \\ X_9 &\to 0 \\ X_{10} &\to 1 \end{aligned}$$

In this grammar, we have the derivation

$$\begin{aligned} S &\Rightarrow X_7X_1 \\ &\Rightarrow 0X_1 \\ &\Rightarrow 0SX_9 \\ &\Rightarrow 0S0 \\ &\Rightarrow 0X_8X_20 \\ &\Rightarrow 01X_20 \\ &\Rightarrow 01SX_{10}0 \\ &\Rightarrow 01S10 \\ &\Rightarrow 01010, \end{aligned}$$

which takes 9 steps, in agreement with our Theorem from class (for a word of length 5).

2. The Connection Between Number of Variables and Long Words

Consider the CNF grammar

$$\begin{aligned} G: S &\to 0|XY \\ X &\to 0|SY \\ Y &\to 1|XS. \end{aligned}$$

A short derivation with no repeated variables is

$$S \Rightarrow XY \Rightarrow 0Y \Rightarrow 01.$$

A longer derivation with repeated variables is

$$S \Rightarrow XY \Rightarrow SYY \Rightarrow 0YY \Rightarrow 01Y \Rightarrow 011.$$

3. The Class of DCFLs Is Closed Under Taking Complements

Theorem 1. *Let Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a DPDA. Then there exists a DPDA M' such that $L(M') = (L(M))'$, in other words,*
The class of DCFLs is closed under taking complements.

Proof. Overview: We need to fix the problems inherent in the intuitive approach of simply swapping the accept and non-accept states from the given DPDA M. In detail, we must ensure that words are accepted by M' (and thereby included in the complement language) which cause M to:

- crash, or
- run forever.

We will therefore start by constructing a new DPDA, M_1, which mimics the behaviour of M, except that M_1

- has only two "sink" final states (accepting and non-accepting), and
- does not crash because of having no transitions or encountering an empty stack.

Then from M_1, we will construct another DPDA, M_2, which mimics the behaviour of M_1, except that M_2

- has no infinite ε-loops.

Then we will be able to swap the accept and non-accept states of M_2, to obtain the desired DPDA M'.
Construction of M_1: Let

$$M_1 = (Q \cup \{q_{-1}, r', f'\}, \Sigma, \Gamma \cup \{X_0\}, \delta_1, q_{-1}, X_0, \{f'\}).$$

Construct δ_1 from δ as follows.

- Keep all existing transitions from δ.
- Ensure that f' is the unique (sink) accept state and r' is the unique (sink) reject state in M_1. Accordingly, add transitions
  - $\delta_1(r', x, *) = (r', *), \forall x \in \Sigma$,
  - $\delta_1(q, \varepsilon, *) = (r', *), \forall q \in Q \setminus F$ where this is allowed for a DPDA,
  - $\delta_1(f', x, *) = (f', *), \forall x \in \Sigma$, and
  - $\delta_1(f, \varepsilon, *) = (f', *), \forall f \in F$ where this is allowed for a DPDA.
- Ensure that M_1 never crashes because of no outgoing transition. Add the missing outgoing transitions for all states, input letters and top-of-stack symbols (preserving the top of stack symbol in every case). Make the new state r' the target. This keeps threads which crash in M because of no outgoing transition alive and in a non-accept state in M_1 until the end of input is reached.

- Ensure that M_1 never crashes because of empty stack. This part of the construction is analogous to the constructions from class showing that acceptance by final state and acceptance by empty stack are equivalent. Add transitions to the existing transitions from M:
  - $\delta_1(q_{-1}, \varepsilon, X_0) = (q_0, Z_0X_0)$ (prime the stack to detect empty stack crashes in M)
  - $\delta_1(q, x, X_0) = (r', X_0), \forall x \in \Sigma, q \in Q$ (words which caused an empty stack crash in M are now explicitly rejected in M_1)
  - $\delta_1(q, \varepsilon, X_0) = (r', X_0), \forall q \in Q$ (similar)

This completes the construction of M_1.
Construction of M_2: A spurious transition in M_1 is a transition of the form

$$(p, \varepsilon, X) \to (q, \gamma)$$

such that

$$(p, \varepsilon, X) \overset{*}{\vdash} (p, \varepsilon, X\alpha),$$

for some stack contents α (possibly empty).
Note that these spurious transitions can be identified in finitely many steps, because all ingredients in the definition of the machine M (and therefore of M_1) are finite, and therefore we only need to follow finitely many ε-transitions at each step.
Remove all spurious transitions. If $(p, \varepsilon, X) \to (q, \gamma)$ is a spurious transition, then replace it with

- $(p, \varepsilon, X) \to (r', X)$ if $p \neq f'$.

Note that the accept state f' has already been handled by the construction above.
Now observe that

- All infinite loops involve a spurious transition. (Think about the stack height as an infinite loop runs, examine the infinitely many local minima, as the construction is finite, there must exist a repetition.)
- Deleting spurious transitions as described above does not change the language of the machine.

This completes the construction of M_2.
Finishing the proof: Now having constructed M_2 as required, we can construct M' by swapping the accept and non-accept states of M_2, completing the proof. ☐

4. DCFLs Are Not Closed Under Unions / Intersections

Define

$$\begin{aligned} L_{abc} &= \{a^n b^n c^n \mid n \geq 0\} \\ L_1 &= \left\{a^i b^j c^k \mid i, j, k \geq 0\right\} \\ &= L\left(a^* b^* c^*\right) \\ L_2 &= \left\{a^i b^i c^k \mid i, k \geq 0\right\} \\ L_3 &= \left\{a^i b^k c^k \mid i, k \geq 0\right\} \end{aligned}$$

We proved in class that L_{abc} is not context-free. Further observe that

(1) L_1 is a regular language, and hence a DCFL,
(2) L_2 is a DCFL, and
(3) L_3 is a DCFL.

Further, we have that

$$(L_{abc})' = (L_1)' \cup (L_2)' \cup (L_3)'.$$

(Each of L_1', L_2' and L_3' is one way that a word can fail to be in L_{abc}.)
Since DCFL is closed under complementation, and CFL is closed under union, it follows that $(L_{abc})'$ is a context-free language. However $(L_{abc})'$ 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 also closed under union.

5. Does There Exist a Non-Context-Free Language In Which All Long Words Can Be Pumped?

Yes. Work out the following problem at your leisure.

(1) Let $\Sigma = \{a, b, c, d\}$.
 (a) Prove that $L = \{ab^jc^jd^j \mid j \geq 0\}$ is **not** a context-free language.
 (b) Prove that $F = \{a^ib^jc^kd^\ell \mid i, j, k, \ell \geq 0 \text{ and if } i = 1 \text{ then } j = k = \ell\}$ is **not** a context-free language.
 (c) Exhibit with proof a choice of a positive integer n, such that, for any $z \in F$, we may write $z = uvwxy$ where
  (i) $|vwx| \leq n$,
  (ii) $|vx| \geq 1$ and
  (iii) $uv^iwx^iy \in F$, for all $i \geq 0$.