ECS120 Introduction to the Theory of Computation
Fall Quarter 2007

Homework 5 Help
Due Monday November 05, 2007

Problem 5.1

Find a decision procedure which determines if a given CFG (with alphabet $\{ a, b \}$) accepts at least one string which contains exactly 4 b's. You can assume that you have procedures that can convert PDAs into CFGs and CFGs into PDAs.

A decision procedure is an algorithm/procedure that provides a yes or no answer for a decision problem. The input to this decision procedure is a context-free grammar G, and should output as follows:

$$\textsc{Accept4B}(G) = \begin{cases} \texttt{yes} & \text{if } G \text{ accepts a string with exactly 4 } b\text{'s} \\ \texttt{no} & \text{otherwise} \end{cases}$$

Keep in mind that G may accept an infinite number of strings. Decision procedures must be bounded. If a procedure may loop infinitely, there may be some cases where it would never output an answer.

Hint: Can you create a language that only accepts strings with exactly 4 b's?

Problem 5.2

Prove the following about context-free languages:

(a) Prove that context-free languages are closed under the $*$ operation (Kleene closure).

(b) Assume that you know that $L = \{ a^n b^n c^n \mid n \geq 0 \}$ is not context-free. Prove that context-free languages are not closed under intersection.

(c) Assuming the same as in (b) above, show that CFLs are not closed under complement.

To show that an operation is closed under context-free languages, assume you have a grammar G and show how to build a new grammar G' such that $L(G') = L(G)^*$. Try to include an argument that G' does actually represent $L(G)^*$.

To show that an operation is **not** closed under context-free languages (part b and c), use proof by counterexample. Find two context-free languages L_1 and L_2 such that the operation results in the language L. (For example, find languages such that $L_1 \cap L_2 = L$.) Since L is not context-free (given), then the operation is not closed under context-free languages.

Hint: Can you rewrite $L_1 \cap L_2$ using complements?

Problem 5.3

> Let $D = \{ xy \mid x, y \in \{0,1\}^* \text{ and } |x| = |y| \text{ but } x \neq y \}$. Show that D is a context-free language.
> (Sipser Problem 2.23)

This is a star-problem in your book, so expect it to be difficult.

The definition on page 101 states "Any language that can be generated by some context-free grammar is called a *context-free language*." Theorem 2.20 on page 115 states "A language is context free if and only if some pushdown automaton recognizes it." You'll be able to do this problem similar to ones on regular languages, except this time with CFGs and PDAs instead of DFAs and NFAs.

Problem 5.4

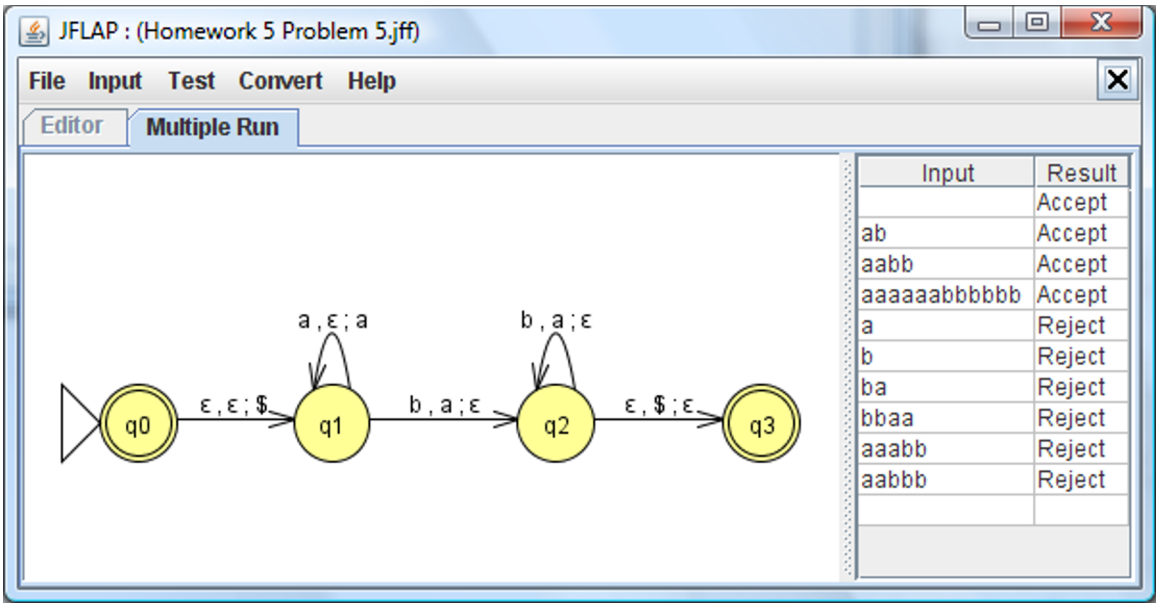
> Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.
>
> $$\begin{aligned} A &\rightarrow BAB \mid B \mid \epsilon \\ B &\rightarrow 00 \mid \epsilon \end{aligned}$$
>
> (Sipser Problem 2.14)

See the discussion notes or your book for the procedure. Follow it exactly!

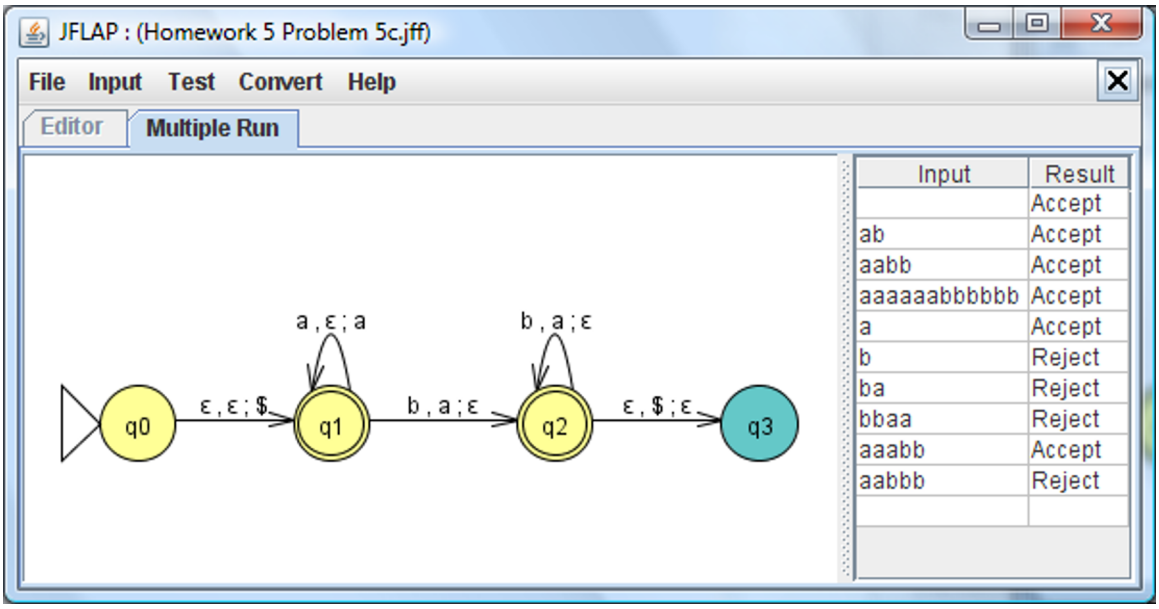
Problem 5.5

> Show that the complement of the language $L = \{ a^n b^n \mid n \geq 0 \}$ is context-free by designing a push-down automaton that recognizes it.

You might want to start by creating a PDA P such that $L(P) = L$:



Will switching the accept and non-accept states give us a PDA P' such that $L(P') = \overline{L}$?



Not even close!

For a DFA M, we can easily find $\overline{L(M)}$ by switching the accept and non-accept states in M. However, we can't do this for an NFA. Why is that? Nondeterminism complicates taking the "complement" of an automaton. Is it possible to create a deterministic PDA? Will we be able to complement that easier?