University of Virginia - cs3102: Theory of Computation Spring 2010

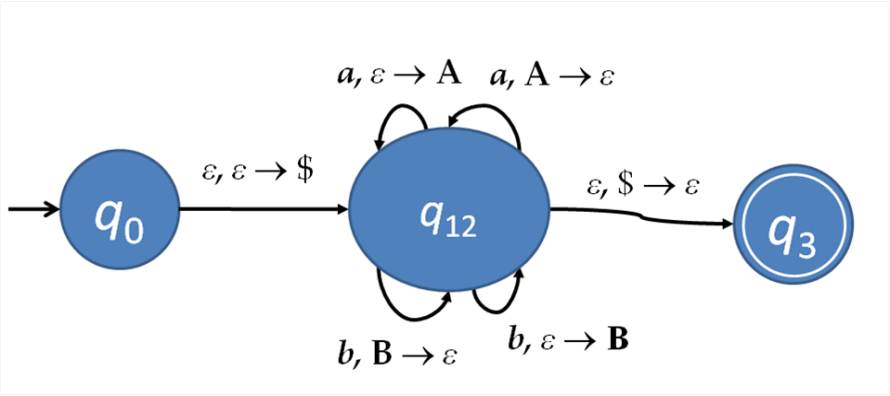
PS3 **Due: 23 February 2010 (2:00pm)**

This problem set covers material through the end of Chapter 2. Answer the first five questions and optional bonus question 6. For full credit, answers must be concise, clear, and convincing, not just correct. Note that you only have *one week* for this problem set, unlike the first two problem sets. Hence, it is designed to be somewhat shorted than PS1 and PS2, but includes questions to give you an opportunity to practice solving problems on the topics that will be on Exam 1 than have not been covered by the first two problem sets.—

You are strongly encouraged to use LaTeX to produce your submission, and we have provided a LaTeX template to assist with this. You may, however, handwrite your answers *so long as your writing is legible and easily interpreted.* For full credit, answers must be concise, clear, and convincing, not just correct.

Collaboration Policy. For this assignment, we will follow the same collaboration and resource policy as on Problem Set 1 and Problem Set 2. See the Problem Set 1 handout for a full description of the policy. Please keep in mind, though, that for the Exam you will need to solve problems on your own. Use collaboration to help you learn, but not in ways that prevent you from learning to solve problems on your own.

Problem 1: Mystery Language. In Class 7, we considered how the language would change if states q_1 and q_2 from the NPDA on slide 7 were merged, producing this NPDA:



Describe precisely the language accepted by this nondeterministic PDA.

> **Challenge Bonus.** Is there a *deterministic* PDA that recognizes the same language? Include a convincing proof that supports your answer.

Problem 2: NPDAs and CFGs. Describe a NPDA that recognizes the language produced by this context-free grammar:

$$
\begin{array}{lcl}
S & \rightarrow & 1A \mid A1 \mid 0B \mid B0 \mid \epsilon \\
A & \rightarrow & 0S1 \mid 1S0 \mid CS \mid SC \\
B & \rightarrow & S11 \mid 1S1 \mid 11S \\
C & \rightarrow & 01 \mid 10
\end{array}
$$

Problem 3: Context-Free Grammars. Provide a context-free grammar that recognizes the language:

$$\{w | w \in \{a,b\}^* \wedge w \text{ contains more } a\text{s than } b\text{s.}\}$$

Use as few nonterminals as possible.

Problem 4: Priming the Pump Redux. Prove the language, $PRIMES$, is not context-free:

$$PRIMES = \{1^n | n \text{ is a prime number}\}$$

Problem 5: Closure Properties.

a. Prove that the context-free languages are closed under concatenation.

b. Prove that the intersection of a context-free language with a regular language is always a context-free language.

c. Prove that the context-free languages are **not** closed under intersection.

Problem 6: Parsing. Below is a slightly simplified excerpt from the actual Java grammar specification (from http://java.sun.com/docs/books/jls/third_edition/html/syntax.html#18.1, Chapter 18). I have changed the syntax to match the context-free grammar notation used in Sipser and the class.

Expression	$\rightarrow$	*Expression1 OptAssignmentOperator*
OptAssignmentOperator	$\rightarrow$	ϵ | *AssignmentOperator Expression1*
Expression1	$\rightarrow$	*Expression2 OptExpression1Rest*
OptExpression1Rest	$\rightarrow$	ϵ | *Expression1Rest*
Expression1Rest	$\rightarrow$	**?** *Expression* **:** *Expression1*
AssignmentOperator	$\rightarrow$	**=**
Expression2	$\rightarrow$	*Expression3 OptExpression2Rest*
OptExpression2Rest	$\rightarrow$	ϵ | *Expression2Rest*
Expression2Rest	$\rightarrow$	*InfixExpressionList*

InfixExpressionList	→	ϵ | *InfixExpression InfixExpressionList*
InfixExpression	→	*InfixOp Expression3*
InfixOp	→	|| | **&&** | **==** | **+**
Expression3	→	*Primary SelectorList*
Primary	→	**(** *Expression* **)** | **Identifier** | **Literal**
SelectorList	→	ϵ | *Selector SelectorList*
Selector	→	**[** *Expression* **]** | **. Identifier**

The terminal **Identifier** is any valid Java identifier (see Section 3.8 of the Java Language Specification for the grammar for Identifiers) and **Literal** is any literal.

a. Consider the following Java expression:

```
true ? false ? true == true : false : false == false
```

which evaluates to `false`. The Boolean values `true`, and `false` are **Literal**s. The conditional expression, $Expression_{pred}$ **?** $Expression_{consequent}$ **:** $Expression_{alternate}$, is evaluated by first evaluating $Expression_{pred}$, which must evaluate to a boolean. If it evaluates to true, then the value of the conditional expression is the value obtained by evaluating $Expression_{consequent}$ (and $Expression_{alternate}$ is not evaluated). If it evaluates to false, then the value of the conditional expression is the value obtained by evaluating $Expression_{alternate}$ (and $Expression_{consequent}$ is not evaluated).

By adding only parentheses, transform it into a grammatical Java expression that evaluates to `true`.

b. Explain how to change the grammar rules so the original expression in the previous part evaluates to `true`. Your new grammar should produce exactly the same language as the original grammar. It is acceptable if your answer leads to an ambiguous grammar, as long as one possible parse of the expression in your grammar evaluates to `true`. (**Challenge Bonus.** Produce an unambiguous grammar for this question. It should produce exactly the same language as the original grammar, but the only possible parse of the expression evaluates to `true`.)

End of Problem Set 3.

Turn in your *stapled* submission at the beginning of class on Tuesday, 23 February 2010.