

Summer 2008 Midterm Solution

Your Name: _____

SUNet ID: _____

Instructions:

- **Honor.** You must complete this exam alone and in the time specified. You must follow the Stanford Honor Code.
- **Time Limit.** You have from 7:00pmPDT to 8:30pmPDT to complete this exam.
- **Show and explain your work.** Do this whenever possible. It allows us to better characterize your understanding of the material - and better grade your hard work.
- **Notes.** You may refer to any books, notes, or other printed material which you bring to the exam. You must *not* use any electronic materials or equipment.
- **DFA transitions.** In the interest of time, you may omit transitions for any DFA; such missing transitions will be assumed to go to an implicit reject state.
- **Submission for Remote Students.** Submit your work electronically via e-mail to dgu@stanford.edu. You may alternatively submit it via fax. You must submit your work immediately after the 8:30pmPDT deadline.
- **Questions During the Exam.** I will be outside the exam room in case you have a question. Remote students may call 304-541-2399 to ask a question.

Problem #	Points Earned	Points Possible
1		6
2		16
3		20
4		12
5		6
6		15
Total		75

Problem 1 File Extensions Considered Harmful

Background. In UNIX, the `sed` utility can be used for transforming text which matches a regular expression. Specifically, `sed` can replace an all occurrences of a regular expression `RE` with some new text `TXT`. This is the format of the `sed` command:

```
sed -e 's#RE#TXT#g' INPUT_FILE
```

For example, to use `sed` to get rid of the ugly carriage returns (`\r`) from a DOS file, you could do the following to replace them with the empty string:

```
sed -e 's#\r##g' INPUT_FILE
```

Problem. Assume you have an input file which contains a large list of file names with one file name per line. **Write a compact regular expression to match file extensions** which you could use as the `RE` in the above command to remove all file extensions. Use *lex regular expression syntax* to specify your expression. Make the following assumptions:

- A file extension is a period followed by zero or more symbols except periods and forward slashes.
- A file extension only occurs at the end of a string.

Example Input

```
p1.tex
/home/dgu/ChessMeister.cpp
/usr/class/PP-1.tar.gz
/dev/nf2
/dev/null/pp3/.gitignore
```

Example Output

```
p1
/home/dgu/ChessMeister
/usr/class/PP-1.tar
/dev/nf2
/dev/null/pp3/
```

Hint: You are only writing a regular expression to match the *file extension* - it should not match any other part of the file name.

SOLUTION: `\.[^\./\n]*$`

Problem 2 Nothing Ever Changes

- a. Remove any λ -cycles from the N DFA shown in Figure 1.

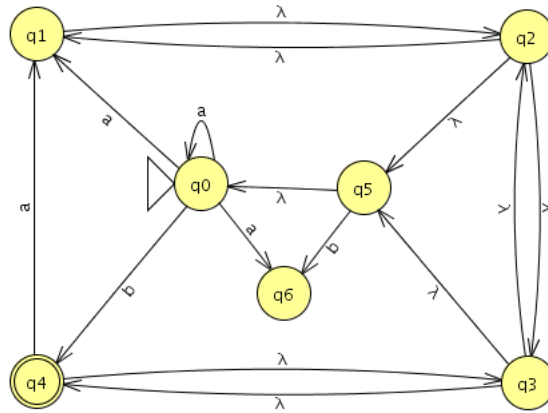
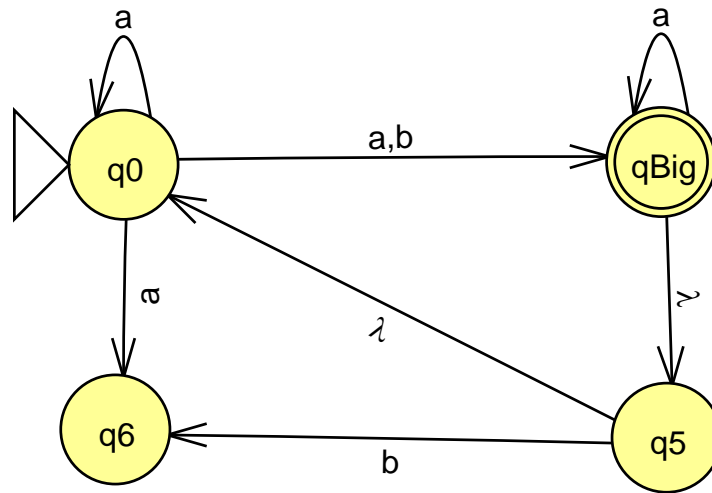


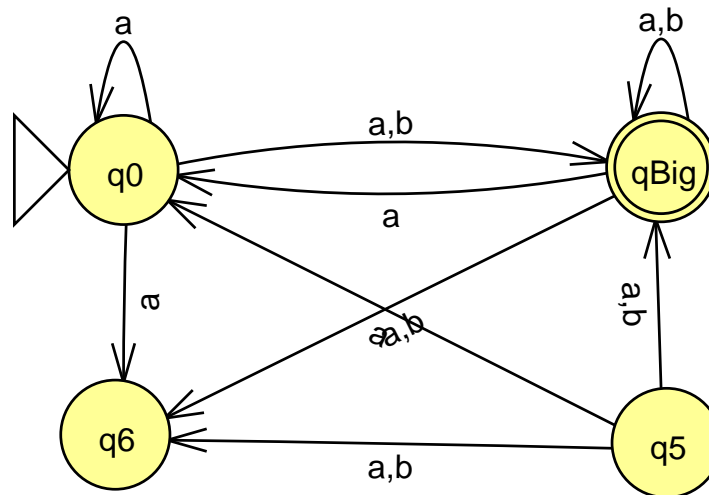
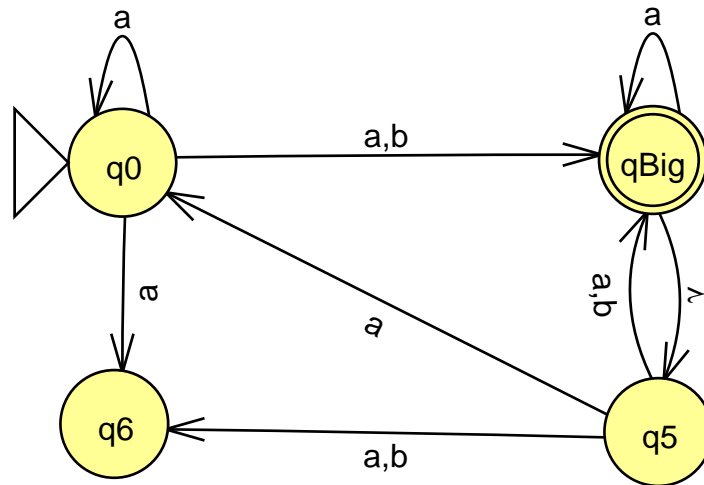
Figure 1: The N DFA to remove λ cycles and transitions from.

SOLUTION: The λ -cycle in this N DFA consists of states $q_1, q_2, q_3,$ and q_4 because the state machine can move freely back and forth between these states without consuming any input.



- b. With the N DFA resulting from the previous step, remove any remaining λ transitions. Remove one λ transition at a time, and show *each* resulting N DFA.

SOLUTION: As several students pointed out, state q_5 is pointless now since it has no transitions going into it and is not the start state. Furthermore, q_6 is also a bit pointless - it has transitions in, but no transitions out and is a reject state. Answers which included any combination of these states with the appropriate transitions were given full credit.



- c. Algorithmically remove any non-determinism from the NFA shown in Figure 2. Show your work with a table, and do not try to simplify the NFA. You do *not* need to draw the resulting DFA.

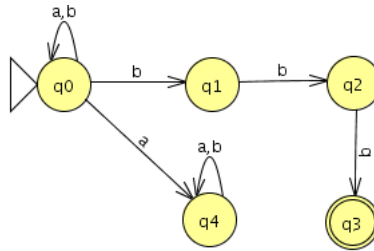


Figure 2: The NFA to remove non-determinism from in order to create an equivalent DFA.

SOLUTION:

State	Symbols		Accept State?
	a	b	
{q0}	{q0q4}	{q0q1}	No
{q0q4}	{q0q4}	{q0q1q4}	No
{q0q1}	{q0q4}	{q0q1q2}	No
{q0q1q4}	{q0q4}	{q0q1q2q4}	No
{q0q1q2}	{q0q4}	{q0q1q2q3}	No
{q0q1q2q4}	{q0q4}	{q0q1q2q3q4}	No
{q0q1q2q3}	{q0q4}	{q0q1q2q3}	Yes
{q0q1q2q3q4}	{q0q4}	{q0q1q2q3q4}	Yes

d. Algorithmically minimize the DFA shown in Figure 3. Show your work with table(s). Draw the resulting DFA.

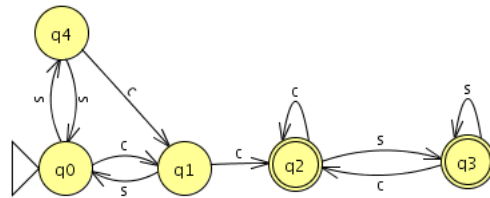
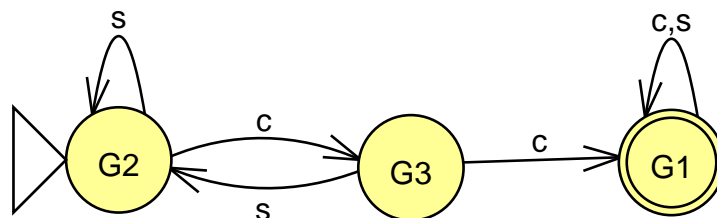


Figure 3: The DFA to minimize.

SOLUTION:

Group	State	c	s
G1	q2	G1	G1
	q3	G1	G1
G2	q0	G2	G2
	q1	G1	G2
	q4	G2	G2

Group	State	c	s
G1	q2	G1	G1
	q3	G1	G1
G2	q0	G3	G2
	q4	G3	G2
G3	q1	G1	G2



Problem 3 How Much Wood ... ?

Consider the following grammar over the tokens {if, wood, could, chuck, a, howmuch}:

$$S \rightarrow Q \text{ OptCond} \mid \text{Stmt OptCond}$$
$$\text{OptCond} \rightarrow \text{if Stmt} \mid \epsilon$$
$$\text{Stmt} \rightarrow N V \text{ OptObj}$$
$$V \rightarrow \text{wood } V \mid \text{could } V \mid \text{chuck}$$
$$\text{OptObj} \rightarrow N \mid \epsilon$$
$$N \rightarrow \text{a wood chuck} \mid \text{wood}$$
$$Q \rightarrow \text{howmuch wood wood Stmt} \mid \text{could Stmt}$$

This grammar allows us to generate sentences such as "howmuch wood wood a woodchuck chuck if a woodchuck could chuck wood."

- The TAs verified that this grammar is unambiguous using a piece of software you should be familiar with. In one sentence, how do you think they did this?

SOLUTION: They ran it through Bison and found that there were no shift/reduce conflicts, proving that the grammar is LALR.

- In one sentence, what are the drawbacks to the method proposed above?

SOLUTION: Not all unambiguous grammars are LALR, so this method will reject many correct grammars.

- Without modifying the grammar, fill in the following LL(1) parse table on the following page.

SOLUTION:

	wood	could	chuck	a	howmuch	if	\$
S	S \rightarrow Stmt OptCond	S \rightarrow Q Cond		S \rightarrow Stmt OptCond	S \rightarrow Q Cond		
OptCond						OptCond \rightarrow if Stmt	OptCond $\rightarrow \epsilon$
Stmt	Stmt \rightarrow N V OptObj			Stmt \rightarrow N V OptObj			
V	V \rightarrow wood V	V \rightarrow could V	V \rightarrow chuck				
OptObj	OptObj \rightarrow N			OptObj \rightarrow N		OptObj $\rightarrow \epsilon$	OptObj $\rightarrow \epsilon$
N	N \rightarrow wood			N \rightarrow a wood chuck			
Q		Q \rightarrow could Stmt			Q \rightarrow how- much wood wood Stmt		

Problem 4 Broken Calculator

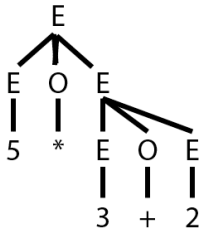
Assume each production in the following grammar is associated with the correct semantic action that computes its value in bison. *Please write your answers in the right margin.*

$$E \rightarrow E O E \mid T_IntConst$$

$$O \rightarrow '+' \mid '-' \mid '*' \mid '/'$$

- a. If no precedence is specified, what value will Bison compute for the string $5*3+2$? Draw the parse tree.

SOLUTION: 25. Bison defaults to shift, so it will read the $+$ rather than reducing the $5*3$.



- b. Draw the parse tree and specify the value it computes for the string $5*3+2$ if we provide only the following precedence directive:

```
%left * /
```

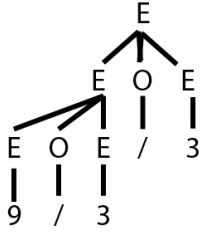
SOLUTION: This will result in the same value and parse tree. It doesn't specify the precedence of the $*$ operator relative to the $+$ operator.

- c. Draw the parse tree and specify the value it computes for the string $5*3+2$ if we provide only the following precedence directive:

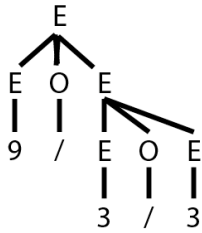
```
%left * / %left + -
```

SOLUTION: This will result in the same value and parse tree. The only difference is that Bison will no longer need to resolve any shift-reduce conflicts automatically. The precedence directives tell it unambiguously to make this tree.

- d. If we specify that division is left-associative, what is the value of $9/3/3$? Draw the parse tree.

SOLUTION: 1.

- e. If we specify that division is right-associative, what is the value of 9/3/3? Draw the parse tree.

SOLUTION: 9.

- f. Draw the parse tree and specify the value it computes for 9/3/3 if we give the following precedence directive:

```
%nonassoc /
```

SOLUTION: The chaining of the / operators is a syntax error when they have been specified not to associate. Without special error recovery code, Bison would not finish parsing the string and would not compute a final value.

Problem 5 The Infamous Alderson Loop

Rewrite the following grammar to be LL(1):

$$\mathbf{A} \rightarrow \mathbf{A}bb \mid \mathbf{A}bc \mid cc$$

SOLUTION: This grammar has both left-recursion and ambiguity - we must remove both for the grammar to become LL(1). There are a number of ways to arrange this grammar to be LL(1), but here is one answer:

$$\mathbf{A} \rightarrow cc\mathbf{A}'$$

$$\mathbf{A}' \rightarrow b\mathbf{X}\mathbf{A}' \mid \lambda$$

$$\mathbf{X} \rightarrow b \mid c$$

Problem 6 Simple Lists

Draw the LR(0) configurating sets for the following grammar. If there are any problems in the grammar, identify where they occur when constructing the configurating sets.

$$\text{NT} = \{ \mathbf{D}, \mathbf{P}, \mathbf{S}, \mathbf{T} \}$$

$$\text{T} = \{ (,), \text{id}, \text{int}, \text{void} \}$$

$$\mathbf{S} \rightarrow (\mathbf{P})$$

$$\mathbf{P} \rightarrow \mathbf{D} \mathbf{P}$$

$$\mathbf{P} \rightarrow \mathbf{D}$$

$$\mathbf{D} \rightarrow \mathbf{T} \text{id}$$

$$\mathbf{T} \rightarrow \text{void}$$

$$\mathbf{T} \rightarrow \text{int}$$

SOLUTION: Assuming rules are numbered from one to six in the order listed, the following configuring sets would be generated:

I_0 :

$S' \rightarrow \bullet S \quad I_1$
 $S \rightarrow \bullet (P) \quad I_2$

I_1 :

$S' \rightarrow S \bullet \quad \text{accept}$

I_2 :

$S \rightarrow (\bullet P) \quad I_3$
 $P \rightarrow \bullet D P \quad I_4$
 $P \rightarrow \bullet D \quad I_4$
 $D \rightarrow \bullet T \text{ id} \quad I_5$
 $T \rightarrow \bullet \text{void} \quad I_6$
 $T \rightarrow \bullet \text{int} \quad I_7$

I_3 :

$S \rightarrow (P \bullet) \quad I_8$

I_4 :

Shift-reduce error

$P \rightarrow D \bullet P \quad I_9$
 $P \rightarrow D \bullet \quad \text{Reduce 3}$
 $P \rightarrow \bullet D P \quad I_4$
 $P \rightarrow \bullet D \quad I_4$
 $D \rightarrow \bullet T \text{ id} \quad I_5$
 $T \rightarrow \bullet \text{void} \quad I_6$
 $T \rightarrow \bullet \text{int} \quad I_7$

I_5 :

$D \rightarrow T \bullet \text{id} \quad I_{10}$

I_6 :

$D \rightarrow \text{void} \bullet \quad \text{Reduce 5}$

I_7 :

$D \rightarrow \text{int} \bullet$ Reduce 6

I₈:

$S \rightarrow (P) \bullet$ Reduce 1

I₉:

$P \rightarrow DP \bullet$ Reduce 2

I₁₀:

$D \rightarrow T \text{id} \bullet$ Reduce 4