Midterm I (Solutions)
CS164, Spring 2001

February 22, 2001

- Please read all instructions (including these) carefully.
- There are 9 pages in this exam and 4 questions, each with multiple parts. Some questions span multiple pages.
- You have 1 hour and 20 minutes to work on the exam.
- The exam is closed book, but you may refer to your two pages of handwritten notes.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. We might deduct points if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

NAME: ______________________________

SID or SS#: ______________________________

Circle the time of your section: 9:00 10:00 11:00 12:00

1:00 2:00 3:00 4:00

Problem	Max points	Points
1	10	
2	25	
3	25	
4	40	
TOTAL	100	

1 Regular Expressions and Finite Automata (10 points)

a) Consider the language over the alphabet $\Sigma = \{a\}$ containing strings whose length is either a multiple of 2 or a multiple of 3 (this includes the empty string). Write a regular expression for this language.

Answer:

$$(a\ a)^* \mid (a\ a\ a\ )^*$$

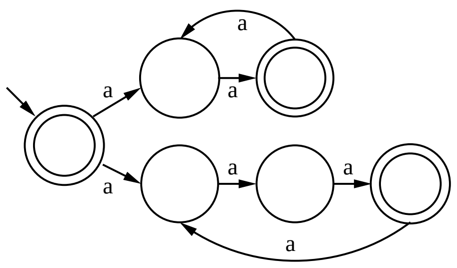
(but as usual for regular expressions there are other equivalent solutions)

Common mistakes:

- $((a\ a) \mid (a\ a\ a\ ))^*$
- Give a CFG instead of a regular expression

b) Draw an non-deterministic finite automaton (NFA) for this language

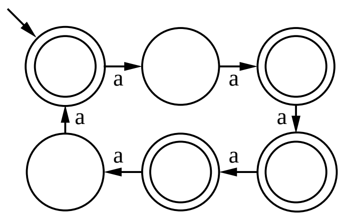
Answer:



Common mistakes:

- Connect the loop arcs to the start state. This gives you a DFA that accepts too many strings (e.g. one of length 5).

c) Draw a deterministic finite automaton (DFA) for this language



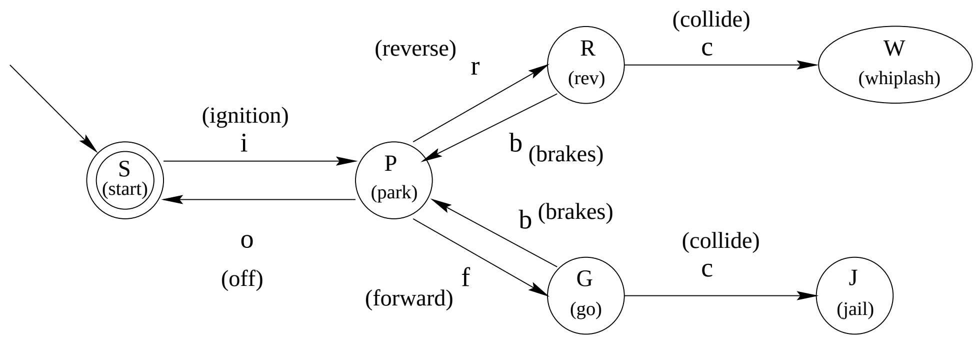
Answer:

Common mistakes:

- Having a state with two outgoing edges labeled "a"
- Label arcs with more than one terminal
- Not having enough states to count up to 6

2 Context-Free Grammars (25 points)

Consider the following DFA over the alphabet $\Sigma = \{b, c, f, i, o, r\}$:



a) (5 points)Write a regular expression that recognizes the same language as the DFA. You may use Flex/JLex notation.

Answer:

$$(i\ (r\ b \mid f\ b)^*\ o)^*$$

(but as usual for regular expressions there are other equivalent solutions)

Common mistakes:

- Not allowing for ϵ to be recognized (Start state is the only accept state.)
- Not accounting for the oi to appear in the internal sequence. Note that the solution allows for strings such as ioirboioifbo.
- Having 'c' as part of an allowable string. (Assuming we accept on whiplash or jail.)
- Not

b) (5 points) Write a context-free grammar that recognizes the same language as the DFA. The grammar can be ambiguous and left-recursive.

Answer:

$$\begin{array}{lcl} S & \rightarrow & i\,T\,o\,S \mid \epsilon \\ T & \rightarrow & r\,b\,T \ \mid \ f\,b\,T \ \mid \ \epsilon \end{array}$$

Common mistakes:

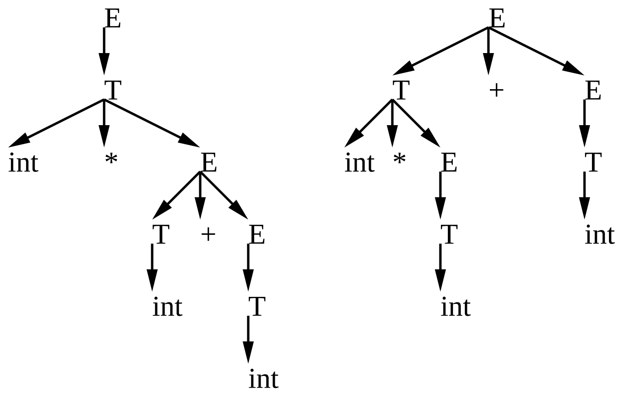
- Using the regular expression given as an answer to part a) as the basis for the grammar rather than referring back to the DFA.
- Not allowing the start non-terminal to derive ϵ.
- Not providing for sequencing of 'oi'.
- Allowing jail and whiplash to be accept states.

c) (7 points) Now consider the grammar:

$$\begin{aligned} E &\rightarrow T \mid T + E \\ T &\rightarrow int \mid int * E \end{aligned}$$

This grammar is ambiguous. Specify an input whose parsing is ambiguous and show two parse trees for it.

Answer: For the input "int * int + int" there are the two parse trees (always the leftmore branch is drawn first):



Common mistakes:

– Most students did well on this problem, though some chose a more complicated example that necessary.

– A few students needed more practice writing out parse trees.

d) (8 ponts) Write an unambigous grammar that recognizes the same language as the grammar in c).

Answer:

$$\begin{aligned} E &\rightarrow T \mid T + E \\ T &\rightarrow int \mid int * T \end{aligned}$$

Common mistakes:

- Left factoring, which leaves the grammar still ambiguous.
- Making it unambiguous but not recognizing the same language. Eliminating simple input such as int * int.

3 LL Parsing (25 points)

Consider the grammar:

$$\begin{aligned} A &\rightarrow B\,C\,D \\ B &\rightarrow h\,B \mid \varepsilon \\ C &\rightarrow C\,g \mid g \mid C\,h \mid i \\ D &\rightarrow A\,B \mid \varepsilon \end{aligned}$$

a) (15 points) Fill-in the table below with the sets *First* and *Follow*. Remember to put ϵ in the set *First*(X) whenever X can reduce to the empty string.

	First	*Follow*
A	$h,\ g\ ,i$	$h,\ \$$
B	$h,\ \epsilon$	$g,\ h,\ i,\ \$$
C	$g,\ i$	$g,\ h,\ i,\ \$$
D	$h,\ g,\ i,\ \epsilon$	$h,\ \$$

Common mistakes:

- Put ϵ in Follow
- Put ϵ in First(A)
- Forget g, i from First(A) and First(D)
- Forget h from Follow(B) (very frequent)
- Put random stuff in Follow

b) (10 points) Fill in the rows for non-terminals B and C in the predictive LL(1) parsing table for the above grammar. Expect to have several entries with more than one production.

	g	h	i	\$
B	ϵ	ϵ | hB	ϵ	ϵ
C	Cg | Ch | g		Cg | Ch | i	

Common mistakes:

- Forget ϵ in [B,h]. This was typically in the same exams where h ws not put in Follow(B). We did not penalize twice for this.
- Put ϵ in the row of C
- Forget Cg and Ch in the cells [C,g] and [C,h]

4 LR Parsing (40 points)

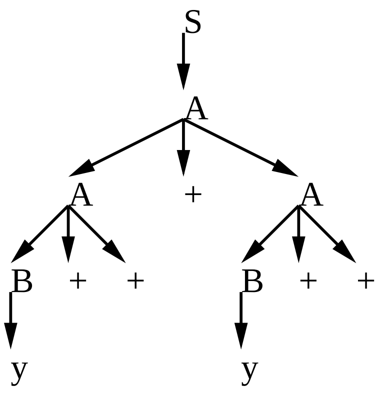
Consider this grammar (the first rule is the conventional start production for LR parsers; you don't need to add another one):

$$\begin{aligned} S &\rightarrow A \\ A &\rightarrow A + A \mid B + + \\ B &\rightarrow y \end{aligned}$$

(Each '+' is a separate token.)

a) (4 points) Draw a parse tree for the input "y + + + y + +"

Answer:



Common mistakes:

- No S at the top of the tree. However, since we said that production was for LR parsing, no points were lost.
- The + was occasionally in place of the A.

b) (6 points) Complete the table below showing the trace of an LR(1) parser on the above input. The "Stack" column must show the stack (with the top at right), the "Input" column shows the not-yet-processed input terminals, and the "Action" column must show whether the parser performs a shift action or a reduce action or accepts the input. In the case of a reduce action say which production is used. We will not grade the contents of the "Input" column and you can use that space as you may see fit.

Stack(with top at right)	Input	Action
	▶ y + + + y + + $	shift
y	▶ + + + y + + $	reduce $B \to y$
B	▶ + + + y + + $	shift
B +	▶ + + y + + $	shift
B + +	▶ + y + + $	reduce $A \to B + +$
A	▶ + y + + $	shift
A +	▶ y + + $	shift
A + y	▶ + + $	reduce $B \to y$
A + B	▶ + + $	shift
A + B +	▶ + $	shift
A + B + +	▶ $	reduce $A \to B + +$
A + A	▶ $	reduce $A \to A + A$
A	▶ $	reduce $S \to A$
S	▶ $	accept

Common mistakes:

- Forget to do the reduction $B \to y$.
- Forget to write the productions entirely. This is why we parse! Not recording the productions defeats the purpose.
- Top of stack at left.
- Forget to shift.

c) (4 points) Compute the set of items contained in the start state of the item-state DFA. That is, compute:

Closure($[S \to \bullet A,\ \$]$) =

Answer: $\{[S \to \bullet A,\ \$], [A \to \bullet A + A, \$/+], [A \to \bullet B + +, \$/+], [B \to \bullet y, +]\}$

Common mistakes:

- Forget $[S \to \bullet A,\ \$]$.
- Add a lookahead of \$ in $[B \to \bullet y, +]$, that is, writing $[B \to \bullet y, \$/+]$.
- Write S instead of A, i.e. $[S \to \bullet A + A, \$/+]$.
- No lookahead of + in the items.

d) (5 points) One of the states in the item-state DFA consists of the set of items $\{[A \to A + A\ \bullet, \$/+], [A \to A \bullet +A, \$/+]\}$. Starting from this state there is a transition labelled "+". Compute the set of items included in the DFA state reached by this transition. That is, compute:

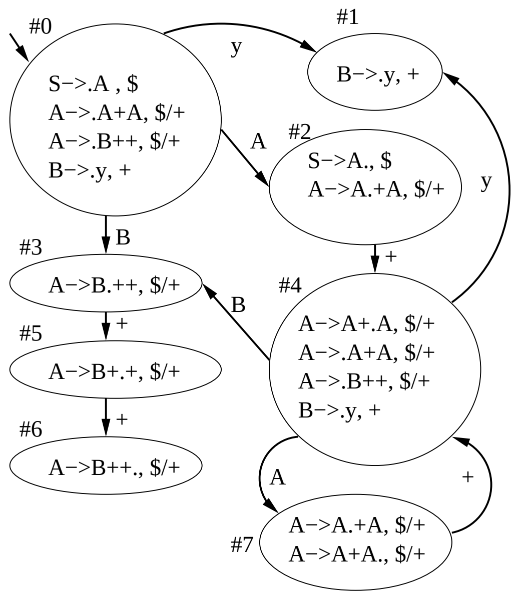
Transition($\{[A \to A + A\ \bullet, \$/+], [A \to A \bullet +A, \$/+]\},\ +$) =

Answer: $\{[A \to A + \bullet A, \$/+], [A \to \bullet A + A, \$/+], [A \to \bullet B + +, \$/+], [B \to \bullet y, +]\}$

Common mistakes:

- Most errors of above.
- No lookahead of \$ in the items.
- Forget $[A \to \bullet A + A, \$/+]$.
- List $[A \to A + A, \$/+]$ with the dot in incorrect locations.

e) (15 pts.) Build the entire item-state DFA for the grammar. Your start state should be as computed at point c) above. Hint: the solution contains 8 states and can be checked against your solution for point b) above.



Common mistakes:

- Put \$ in the lookahead of $[B \to \bullet\ y, +]$.
- Failing to put $[S \to \bullet\ A, \$]$ into the states.
- Forgetting to expand a nonterminal somewhere.

f) (3 pts.) This grammar has a shift-reduce conflict. In which state of the DFA, and on which input symbol, does the conflict occur?

Answer: In the state containing the items $\{[A \to A + A\ \bullet, \$/+], [A \to A \bullet +A, \$/+]\}$ there is a conflict on the input non-terminal $+$.

Common mistakes:

- Pick the wrong state.
- Think that shifting a *non*terminal (a "goto") could cause a conflict.

g) (3 pts.) Assume now that we choose to resolve this conflict in favor of "shift". (That is, when there is a choice of shift or reduce, we shift.)

Specify what is the associativity that corresponds to this choice (circle one):

left [✓ right]

Common mistakes:

- Left.
- Not writing anything (why not guess?).