CS143 Midterm
Spring 2025

- Please read all instructions (including these) carefully.
- There are 5 questions on the exam, some with multiple parts. You have 80 minutes to work on the exam.
- The exam is open note. You may use laptops, phones and e-readers to read electronic notes, but not for computation or access to the internet for any reason other than to access the class webpage.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. Do not write on the back of exam pages or other pages.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. You may get as few as 0 points for a question if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

SUNET ID: _____

NAME: _____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: _____

Problem	Max points	Points
1	20	
2	10	
3	25	
4	30	
5	15	
TOTAL	100	

1. Finite Automatas

(a) Explain why a regular language containing the symbols $\{\varepsilon, a\}$ cannot be recognized by a DFA with exactly one accepting state. Note that the language only contains the two given symbols and not strings (e.g., "aa" is not in the language).

Answer:

Since the language accepts ε, the initial state must also be an accepting state. Since the language accepts "a", there must be an edge "a" from the initial state to an accepting state. Since the initial state is the only accepting state, the edge "a" is from the initial state to itself. So the DFA must also accept string "aa", which is not in the language.

(b) Given any NFA with more than one accepting states, explain how you can construct an NFA with exactly one accepting state that recognizes the same language.

Answer:

Add a new node q into the NFA, making it the only accepting state. Add an ε edge from all original accepting states in the NFA to node q. The transformed NFA has exactly one accepting state and recognizes the same language.

2. Lexical Analysis

Consider the following flex-like lexical specification:

```
a*b    { print "1" }
ba     { print "2" }
a*ba*  { print "3" }
```

Given the following input string:

abaabbaaabaab

What does the lexer print?

Answer:

31331

abaa	3 by the maximum munch rule
b	1 by the precedence rule
baaa	3 by the maximum munch rule
baa	3 by the maximum munch rule
b	1 by the precedence rule

3. Semantic Actions

Consider the regular expressions over the alphabet of lowercase letters $\{a, b, \ldots, z\}$. Define a syntax-directed translation that computes the set of possible *first* characters of a string. Here are two simple examples:

Example input	Possible start characters
$ab + cd$	$\{a, c\}$
ε	$\{\}$

On the following page is a grammar for regular expressions. Fill in the missing semantic actions. Observe the following constraints in writing your actions:

- `EPSILON` is a terminal representing the reglar expression ε.
- There is a *value* attribute for the `CHAR` terminal whose value is the single lowercase character.
- Compute a boolean attribute *nullable* that records whether the empty string is in the language of a regular expression.
- Compute a set attribute *beginnings* that records the set of characters a regular expression can start with. Use standard set operators ($A \cup B$, $A \cap B$, $A - B$, etc) where appropriate. Express an empty set as $\{\}$ and a set containing one element, e.g. \$1.val, as {\$1.val}.
- Use no other attributes but *val*, *nullable*, and *beginnings*.
- Use no global state—all semantic actions must be equations between attributes of the grammar symbols.
- Use bison notation to refer to symbols and attributes, so for instance `$$.beginnings` is the beginnings attribute for the left-hand side non-terminal and `$2.nullable` is the nullable attribute of the second symbol on the right hand side.

Answer:

```
expr -> EPSILON {
    $$.nullable   = true
    $$.beginnings = {}
}
| CHAR {
    $$.nullable   = false
    $$.beginnings = {$1.val}
}
| expr expr {
    $$.nullable = $1.nullable && $2.nullable
    if $1.nullable {
        $$.beginnings = $1.beginnings U $2.beginnings
    }
    else {
        $$.beginnings = $1.beginnings
    }
} | expr '+' expr  {
    $$.nullable   = $1.nullable || $3.nullable
    $$.beginnings = $1.beginnings U $3.beginnings
}
| expr '*' {
    $$.nullable   = true;
    $$.beginnings = $1.beginnings
}
| '(' expr ')' {
    $$.nullable   = $2.nullable
    $$.beginnings = $2.beginnings

}
```

4. Top-Down Parsing

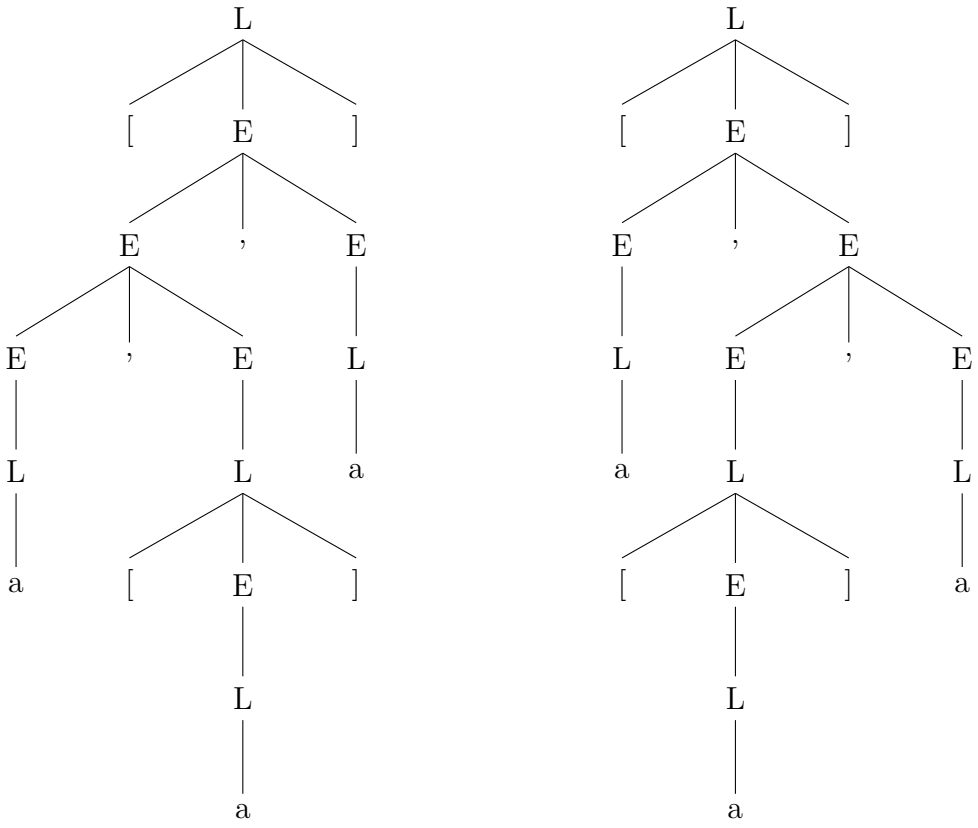
Consider the following grammar G for nested lists:

$L \rightarrow [E] \mid a$
$E \rightarrow E, E \mid L \mid \varepsilon$

where the following symbols are terminals: [ ] , a

(a) This grammar is ambiguous. For the string "$[a, [a], a]$", draw two different parse trees to demonstrate the ambiguity.

Answer:



(b) Provide the First and Follow sets of the non-terminals L and E.

Answer:

First L : { [a }
First E : { , [a ε }

Follow L: { ,] $ }
Follow E: { ,] }

(c) This grammar cannot be parsed using an LL(1) parser due to conflicts. A First-First conflict occurs when a nonterminal has multiple productions and at least one terminal symbol that appears in the First sets of more than one of these productions. Find and explain a First-First conflict in this grammar.

Answer:

The productions $E \to E, E$ and $E \to L$ form a first-first conflict. The tokens '[' and 'a' are present in the first sets of both E and L, so when they appear as the next token there is ambiguity in which production to select.

(d) Eliminate the left recursion from this grammar. Write the complete transformed grammar.

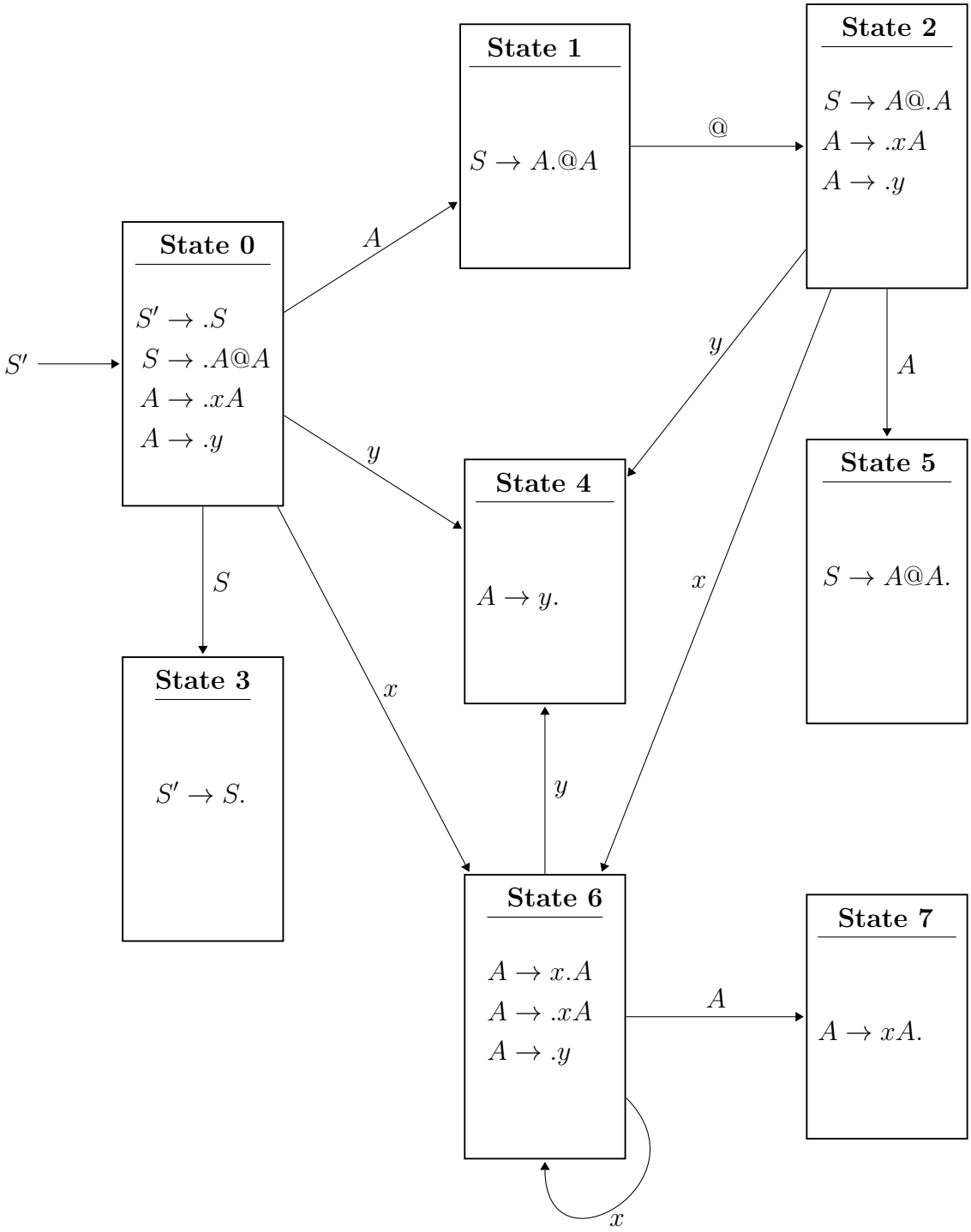
Answer:

One possible answer:

$$
\begin{aligned}
L &\to [E] \mid a \\
E &\to LE' \mid E' \\
E' &\to , EE' \mid \varepsilon
\end{aligned}
$$

5. Bottom-Up Parsing

Consider the following complete DFA for viable prefixes of the context-free grammar on the next page.



Complete the grammar by providing the right-hand sides of the productions for A, so that the grammar is consistent with the LR(0) items.

Answer:

Production

$S' \to S$

$S \to A@A$

$A \to xA$

$A \to y$

(blank page for extended answers)