Midterm I (Solutions)
CS164, Spring 2002

February 28, 2002

- Please read all instructions (including these) carefully.
- There are 9 pages in this exam and 5 questions, each with multiple parts. Some questions span multiple pages. All questions have some easy parts and some hard parts. If you get stuck on a question move on and come back to it later.
- 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#: _____

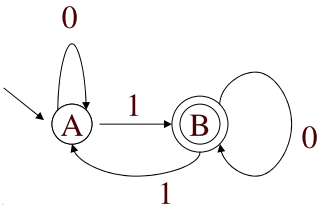
Circle the time of your section: 9:00 10:00 11:00 12:00

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

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

1 Regular Expressions and Finite Automata (20 points)

a) Draw a deterministic finite automaton (DFA) that recognizes the language over the alphabet $\{0, 1\}$ consisting of all those strings that contain an **odd** number of 1's.



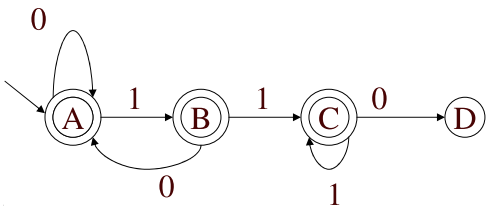
b) Write a regular expression for this language.

Solution:

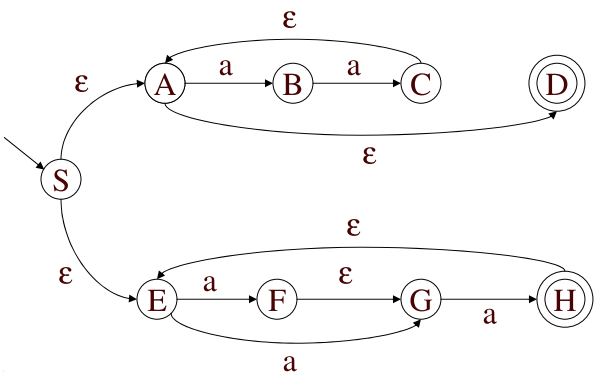
$$(0^*10^*)(10^*10^*)^*$$

c) Draw a deterministic finite automaton (DFA) for the language of all strings over the alphabet $\{0, 1\}$ that do **not** contain the substring 110.

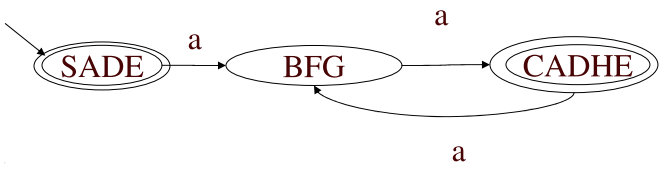
Solution: (state D is a garbage state)



d) Consider the following NFA over the alphabet $\{a\}$. Convert this NFA to a DFA. For each DFA state write the set of the NFA states that it corresponds to.



Solution:



2 Context-Free Grammars (20 points)

The following grammar is ambiguous:

$$E \rightarrow \texttt{id} \mid E + E \mid E * E \mid E \,.\, E \mid E - E$$

a) Show two parse trees for the string `id + id . id`.

b) The ambiguity is removed by adding the following predecence declarations. Recall the convention that the first declaration assigns the lowest precedence:

```
%right + -
%left *
%left .
```

Considering these declarations, show the parse tree for the string:

```
id + id . id . id * id + id
```

d) Write an unambiguous grammar for the same language, resolving ambiguity as specified by the precedence declarations shown above.

Solution:

$$\begin{array}{lcl} E & \rightarrow & T \mid T + E \mid T - E \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & F.id \mid id \end{array}$$

3 LL Parsing (15 points)

a) Consider the following grammar over the alphabet $\{a, b, d\}$:

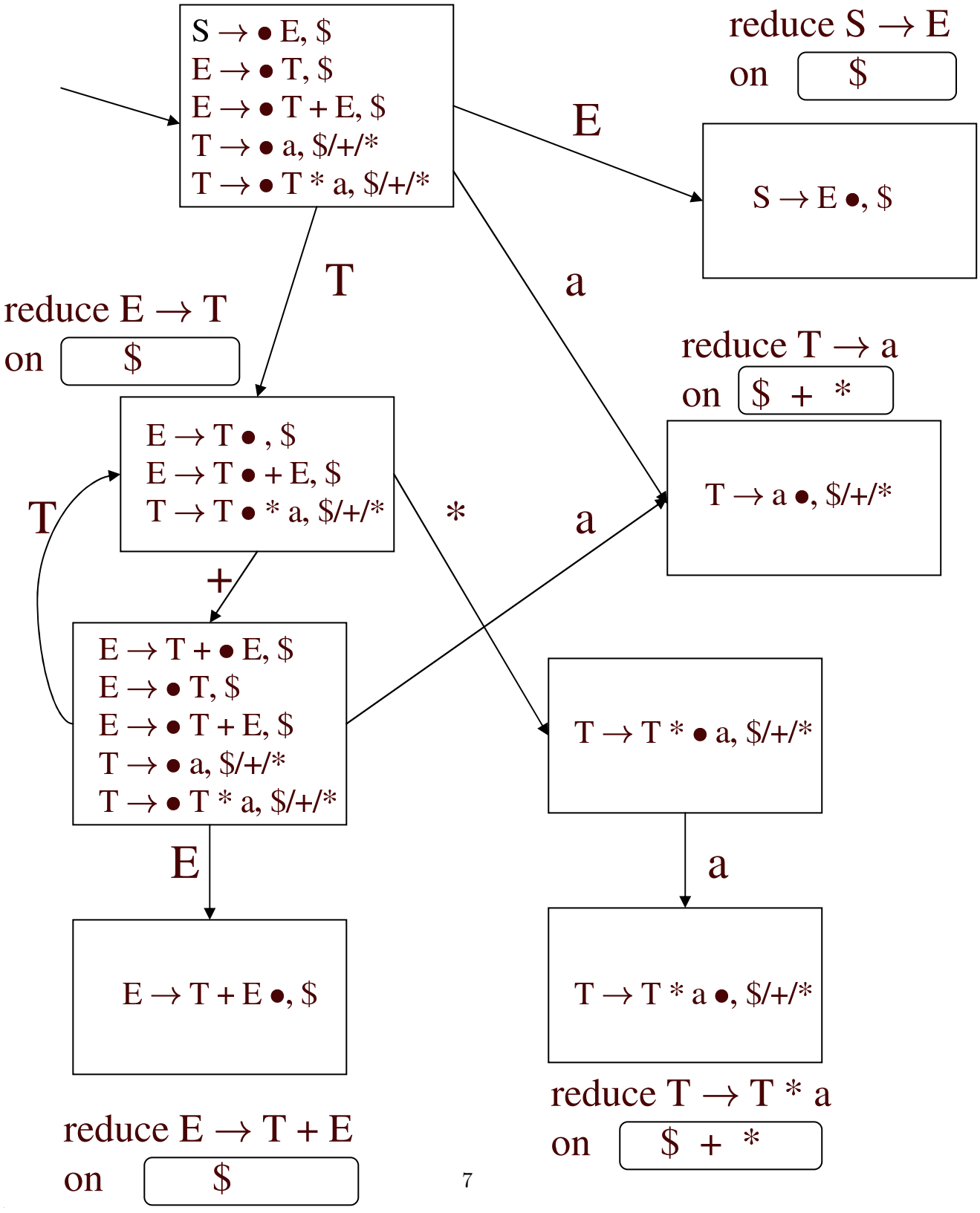
$$\begin{array}{lcl} S & \rightarrow & A\ B\ D \\ A & \rightarrow & a \mid B\ S\ B \\ B & \rightarrow & b \mid D \\ D & \rightarrow & d \mid \epsilon \end{array}$$

Fill in the table below with the First and Follow sets for the non-terminals in this grammar:

	First	Follow
S	a, b, d	b, d, $
A	a, b, d	b, d, $
B	b, d, ϵ	a, b, d, $
D	d, ϵ	a, b, d, $

4 LR Parsing (30 points)

The following figure shows the LR(1) DFA for a grammar. The DFA is missing several elements that you'll have to fill-in: the LR(1) items that describe each state, the labels on the transitions and the lookahead terminals for the reduce labels.



a) Write the grammar that corresponds to the above DFA.
Solution: just copy the productions listed in the reduce labels.

b) Fill in the labels on the transitions. You can do this either by actually filling in the sets of LR(1) items or by examining closely the reduce labels.

c) For each of the following configurations of the LR(1) parser, say what action is taken by the parser (one of "shift" or "reduce with production ..." or "error") and write the next configuration (except in the case of an error).

Configuration	Action	Next configuration
$T \triangleright + a + a\$$	shift	$T + \triangleright a + a\$$
$T + \triangleright * a\$$	error	
$T + T + a \triangleright * a\$$	reduce $T \to a$	$T + T + T \triangleright * a\$$
$T * a \triangleright \$$	reduce $T \to T * a$	$T \triangleright \$$
$T + E \triangleright + a\$$	error	

d) Fill in the large boxes (representing states) with the LR(1) items.

e) Fill in the lookahead terminals in the small boxes next to each reduce label.

5 LL Parsing (15 points)

Consider the following grammar in which S is the start non-terminal and in which the productions for the non-terminals A, B and C are not shown. There are no other productions for S.

$$\begin{array}{lcl} S & \rightarrow & A\ B \mid A\ C \\ A & \rightarrow & \ldots \\ B & \rightarrow & \ldots \\ C & \rightarrow & \ldots \end{array}$$

Consider now the LL(1) parsing table for this grammar. What conditions must be satisfied by the `First` and `Follow` sets of the non-terminals, such that the row corresponding to S in this table **does not** contain multiply-defined entries?

Write your answer in the table below considering six separate cases. For example, in the top-left entry consider the case when $\epsilon \notin \text{First}(B)$ and $\epsilon \notin \text{First}(C)$ and $\text{First}(A) = \{\epsilon\}$, and write what **additional** conditions must be met, or write "never" if no additional conditions would help.

	$\epsilon \notin \text{First}(B)$ $\epsilon \notin \text{First}(C)$	$\epsilon \notin \text{First}(B)$ $\epsilon \in \text{First}(C)$	$\epsilon \in \text{First}(B)$ $\epsilon \in \text{First}(C)$
$\text{First}(A) = \{\epsilon\}$	$\text{First}(B) \cap \text{First}(C) = \emptyset$	$\text{First}(B) \cap \text{First}(C) = \emptyset$ $\text{First}(B) \cap \text{Follow}(S) = \emptyset$	never (since both AB and AC end up in the column for \$)
$\text{First}(A) \neq \{\epsilon\}$	never (since both AB and AC end up in the column for anything in $\text{First}(A) - \{\epsilon\}$)	never	never