CS3300 Midterm

Dept of CSE, IIT Madras
Total marks = 48

September 19, 2014

Read all the instructions and questions carefully. You can make any reasonably assumptions that you think are necessary; but state them clearly. There are total six questions, each 8 marks worth. You will need approximately 20 minutes for answering an 8 marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

You will get an answer sheet with 12 pages (if you get a answer sheet with fewer pages then ask for a replacement sheet). Leave the first page empty. Start each question on a new page. Think about the question before you start writing and write briefly. **The answer for any question (including all the sub-parts) should NOT cross more than two pages.** If the answer for any question is spanning more than two pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page. You mostly would NOT need any additional sheets.

1. [8] **Lexical Analysis**: A language admits the following six operators: $=, +, +=, -=, -$, and $==$.
   Draw a single transition diagram (DFA) for recognizing all these tokens. [2]
   Use the diagram to build the lexical transition table. [2]
   Give a sketch of lexical recognizer that uses this transition table to recognize a series of these tokens delimited by white space. [4]

2. [8] **Parsing I**: State briefly the difference between derivation and reduction, and LL parsing and LR parsing. [2+2]
   Write an example grammar, a sample input string and show the difference between LL parsing and LR parsing. [4]

3. [8] **Parsing II**: Explain briefly why left factoring and left recursion removal are important for LL parsing. [2]
   Briefly state the procedure to do left factoring and left recursion removal (both direct and indirect). [3]
   Apply your procedures on the following grammar. [3]
   `S -> A a | B`
   `A -> A c | S d |` ϵ
   `B -> b | Bb | S`

4. [12] **Parsing III**: Consider the following grammar that generates all possible matched curly braces: $S \rightarrow S$ { S } $S|\epsilon$. Construct the LALR(1) sets of items for the augmented version of the above grammar. [4]
   Build the LALR(1) parsing table. [4]
   State the number of conflicts if/any. [1]
   If conflicts exist, then resolve them in the standard way (favor shift over reduce,

and when the choice is between two productions to reduce, favor the first one). Use the input {} to show the trace of the shift-reduce parser (state of Stack and Input for each Action). [3]

5. [8] **Syntax Directed Definition**: Give an SDD to differentiate mathematical expressions derived from the following grammar.

```
E -> E + T
   | T
T -> T * F
   | F
F -> (E)
   | x // only one specific identifier is allowed
   | num
```

For example, for the input `3 * x * x + 2 * x`, your translation should return `3*2*x+2` or any other equivalent expression (e.g., $3*2*x^1+2*1*x^0$).

6. [4] **Error Detection**: In a compiler, errors can be reported in pretty much every phase. Give two examples each, for the errors that can be reported in the "Lexical Analyzer", "Parser", "Type Checker", and "any later phase". Take languages like MiniJava and MiniIR as the example language and IR.