CS606 Current Mid Term Solved subjective

Date 10-06-2014

Mehranali1991@gmail.com

Linker use in compiler?

A compiler generates object code files (machine language) from source code.A linker combines these object code files into an executable.

Write a regular expression for the language of all words that starts and ends with different letters. Answer:
a(a+b)*b+b(a+b)*a

Diff between/ Left recursion and Left factoring

1. **Left recursion:** when one or more productions can be reached from themselves with no tokens consumed in-between.
2. **Left factoring:** a process of transformation, turning the grammar from a left-recursive form to an equivalent non-left-recursive form.

Bottom-up Parsing

Bottom-up parsing is more general than top-down parsing. Bottom-up parsers handle a large class of grammars. It is the preferred method in practice. It is also called *LR* parsing; *L* means that tokens are read left to right and *R* means that the parser constructs a rightmost derivation. LR parsers do not need left-factored grammars.

Syntactic errors

in computing, an error in a program due to a code that does not conform to order expected by the programming language

Example:

A syntax error occurs when a user (or programmer) has put words in an order that aprogram doe s not understand.

Top down parser cannot be left recursive how we can handle it?
Answer:
Our expression grammar is left recursive. This can lead to non-termination in a top-down parser. Non-termination is bad in any part of a compiler! For a top-down parser, any recursion must be a right recursion. We would like to convert left recursion to right .

Question
What is the role of Automatic Code Generator with respect to compiler?
Answer:
It generates efficient tokens automatically. It is known as Lexer generator.

Question No2
How Linker play an important role with respects to compilers constructions?
A linker or link editor is a computer program that takes one or more source files generated by a compiler and combines them into a single executable program.

Question No 4
Consider the following grammar. Calculate the first set of non- terminal S, A and B

S→ AB
A→ a | ε
B→ b | ε

Solution:

S→ AB

Rules for making first set:

1) If S→ A….Z then first of S will be First of A….Z

2) If First of A….Z all contains ε then first of S will also contain ε

3) If any (single) First set from A….Z doesn't contain ε in their First set then First of S will also doesn't contain ε.

4) The first of A will be first non terminal on r.h.s

5) The first of B will also be the non terminal on r.h.s

6) Note: ε is also a terminal.

Now make first sets for S,A and B

First{A}={a, ε }

First{B}={b ,ε}

Q. Up frontier discovery useful for top down parser.

Answer:
A bottom-up parser builds the parse tree starting with its leaves and working toward its root. The upper edge of this partially constructed parse tree is called its upper frontier

Q. In two pass assembler what is the objective of First Pass? 2 marks
Answer:
First pass contains Scanner + Parser in it. Scanner takes the streams of chcracters as input and converts them into words or tokens
<Token type, word>
While Parser check those tokens for syntactic analysis and semantic analysis if found errors then report them after that it develop an IR.

Q. What is Register allocation? 3 marks
Answer:
Registers in a CPU play an important role for providing high speed access to operands. Memory access is an order of magnitude slower than register access. The back end attempts to have each operand value in a register when it is used. However, the back end has to manage a limited set of resources when it comes to the register file. The number of registers is small and some registers are pre-allocated for specialize used, e.g., program counter, and thus are not available for use to the back end. Optimal register allocation is NP-Complete.

Why bottom up parser is called LR parser?
Answer:
Bottom-up parsing is more general than top-down parsing. Bottom-up parsers handle a large class of grammars. It is the preferred method in practice. It is also called LR parsing; L means that tokens are read left to right and R means that the parser constructs a rightmost derivation. LR parsers do not need left-factored grammars. LR parsers can handle left-recursive grammars.

Best of luck

Remember me in your prays ...!!!!!!