Renat Norderhaug

CS 326

Homework 1

12/11/18

**1. (10 pts) Why are the front end and the back end of a compiler usually implemented as separate passes?**

A pass is defined as a compilation of a phase or square of phases that is serialized with respect to the rest of compilation. The first pass referred to as the “front end” and the second as the “back end” is usually implemented separately between semantic analysis and intermediate code generation or between intermediate code generation and machine-independent code improvement. The front end takes the source code and converts it to intermediate code meanwhile the backend converts the intermediate code to machine code which makes it an easier process with modularity and portability across language and architecture.

1. **(28 pts) For each of the following languages, write a regular expression that describes the language.** 
   1. (7 pts) The set of strings of length three or more, over alphabet {a, b}

(a|b)(a|b)(a|b)(a|b)\*

* 1. (7 pts) The set of natural numbers divisible by 25.

N = 0|1|2|3|4|5|6|7|8|9

(N\*50) | (N (2/7) \*5) | (N \* 0)

* 1. (7 pts) The set of strings that consist of an odd number of a’s, over alphabet {a}.

a(aa)\*

* 1. (7 pts) The set of strings over alphabet {a, b} that begin with at least two a’s, and end with at least two b’s.

aa(a|b)\*bb

1. **(21 pts) For each of the following languages, write a grammar that describes the language.** 
   1. (7 pts) The set of strings over alphabet {a, b} that begin with at least two a’s, and end   with at least two b’s (same language as above).
      * 1. = aaTbb
        2. = |a | b | T | ε
   2. (7 pts) The set of strings that consist of an even number of a’s, over alphabet {a}.

S = aaS | Saa | aSa | ε

* 1. (7 pts) The set of strings of parentheses ( ), brackets [ ], and braces { } that are properly nested. For instance, ( ) [ { } ( ) ] is properly nested, while ( [ ) ] is not.

S = SS | (S) | [S] | {S}| ε

1. **(21 pts) Consider the following grammar for a declaration list:**  decl\_list → decl ; decl\_list | ε decl → specifier type name\_list specifier → const | static | ε type

→ double | int

name\_list → name | name , name\_list name → id args args → ( decl\_list ) | ε

* 1. (4 pts) Indicate whether each of the following strings belongs to the language described by the grammar.

int a (int b); no int c (int d (int e;);); yes double f, g (static int h;); yes static int i; const double j; yes

* 1. (7 pts) Show a leftmost derivation of the string static int f(); under this grammar. static int f(); decl\_list -> delc; delc\_list

-> specific type namelist;

-> static type namelist;

-> static int namelist

-> states int name;

-> static int id args;

-> static int f(delc\_list);

-> static int f();

* 1. (10 pts) Rewrite the grammar so that arguments are separated by commas (similar to the function arguments in C). For instance, each of the following should be a valid string under the new grammar:

int f ();

int f (double x); int f (double x, int y);

decl\_list -> list -> decl; decl\_list | ε | decl; decl\_int | decl

1. **(Extra Credit - 10 pts) Write a grammar that describes the following language: the set of strings that consist of a sequence of a’s followed by a sequence of b’s, where the number of a’s is odd, and equal to the number of b’s, over alphabet {a, b}. In other words, the**

**n n**

**language {a b |n>0 and n is odd}.**s —>id(a) id(b) | id(a)id(a)Sid(b)id(b)