CS 571

Homework 1

Lovelin Anand Edward Paul

B00669954

1. /^ [+-]?(?:(0[0-7]+(\_?[0-7]+)\*) //octal

|(0x[0-9a-e]+(\_?[0-9a-e]+)\*) //hexadecimal

|(0b[0-1]+(\_?[0-1]+)\*) //binary

|([0-9](\_?[0-9]+)\*))$/i //decimal

1. <Digit>

: [0-9]

;

<Sign>

: [+-]

;

<Exponent>

: [eE]

;

<FloatSuffix>

: [dfDF]

;

<SignedInteger>

: <Sign>? <Digit>+

;

<ExponentPart>

: <Exponent> <SignedInteger>

;

<DecimalFloatingPoint>

: <Digit>+ /. <Digit>\* <ExponentPart>?<FloatSuffix>?

| /. <Digit>+ <ExponentPart>?<FloatSuffix>?

| <Digit>+<ExponentPart><FloatSuffix>?

| <Digit>+<ExponentPart>?<FloatSuffix>

;

<HexDigit>

: [0-9a-fA-F]

;

<HexExponent>

: [pP]

;

<HexExponentPart>

: <HexExponent><SignedInteger>

;

<HexPrefix>

: 0[xX]

;

<HexSignifcand>

: <HexPrefix><HexDigit>+

| <HexPrefix><HexDigit>\*.<HexDigit>+

;

<HexDecimalFloatingPoint>

: <HexSignifcand><HexExponent>< FloatSuffix>?

;

<FloatingPointLiteral>

: ^ <DecimalFloatingPoint>

| <HexaDecimalFloatingPoint> $

;

Reference: http://docs.oracle.com/javase/specs/jls/se8/html/jls-3.html#jls-3.10.2

1. a) Whitespaces can be added to the primitive ugly-regexp. With whitespace, primitive ugly-regexp can be redefined as chars (x\_1, …, x\_n) where x\_1, …,x\_n representing any single character. Now it can represent any single whitespace character also.

Whitespaces outside the primitive ugly-regexp’s can be ignored as it does not give any useful information while translating to standard regexp

chars(a,b, ) => [ab\t] // Whitespace is translated

chars(a) + chars(b) => [a]|[b] // Whitespaces are ignored

b) Since whitespace has to be supported, Scanner cannot ignore whitespaces. Scanner does not have any context information to ignore whitespaces selectively. So whitespaces should be added as new token type.

Whitespaces which are present other than ones within primitive ugly-regexp have to be ignored. Instead of making every parse function to ignore whitespaces, we can make check and match function to ignore whitespaces based on the flag received as argument. Based on the ignore whitespace flag, check and match function can ignore or match whitespace tokens.

So parse functions which are checking for non-terminal symbol other than primitive regexp can pass ignore whitespace flag to check and match function while the parse function parsing primitive regexp can check and match whitespaces.

1. These are the possible situations:
2. chars(a) . chars(b) + chars(c) => [a] ( [b] | [c] )

If there are more than one operators with different precedence, then parentheses are applied to the operation of higher precedence

1. ( chars(a) . chars(b) )+ chars(c) => [a] [b] | [c]

If there are more than one operators with different precedence and parentheses are applied to the operation of lower precedence , then the existing parentheses can be removed.

1. \*(chars(a)) => [a]\*

If a Kleene closure is applied to a single ugly Regexp, then there is no need of adding parentheses

1. \*(chars(a) . chars(b)) => ([a][b])\*

If a Kleene closure is applied to more than one ugly Regexp, then a parentheses must be applied to it

So it can be found that,

The non-redundant parentheses depend upon the

1. Number of regexp processed
2. Precedence of operations involved between the regexp
3. Existing parentheses

So, if the number of regexp processed, the precedent operator type and ‘if existing parentheses is present’ flag are passed as argument to each function call, then based on the above possibilities, non redundant expression can be generated. In the same way, each function has to return the same arguments to the caller function to carry on the information to the caller functions

5. Context Free Grammar G for the language *L*

S

: a S b

| b4

;

**Claim:** Every string w belonging to L can be generated by G

**Proof:**

**Base case:**

n = 0, then w = b4, n = 1 , then w = ab4b = ab5

**Inductive hypotheses:**

Assume that the claim is true. For n = k, w = akbk+4 = akb4bk can be generated by G

**Inductive Step**:

Prove that claim for n = k + 1, Let w = ak+1bk+5

With k >= 0, then w = aakbbk+5 = aakb4bkb. = awkb. By the inductive hypotheses, the string akb4bk can be generated by G. Hence proved.

6.

* A grammar consists of non-terminal symbols and terminal symbols.
* A recursive descent parser is a top-down parser which has a parse function for every non - terminal symbol.
* The parse function either leads to another non-terminal symbol or a terminal symbol.
* The recursive descent parser recursively parses the non-terminal symbols until it leads to a terminal symbol. The
* So it is guaranteed that a recursive-descent parser for a grammar amenable to recursive-descent parsing guaranteed to terminate

Void f

7.

b - 0x7ffcd3fe8290

Arguments to called routines

c - 0x7ffcd3fe8298

a - 0x7ffcd3fe829c

t1 - 0x7ffcd3fe82a0

Local Variables/ Temporaries

t3 - 0x7ffcd3fe82a4

t2 - 0x7ffcd3fe82a8

p1 - 0x7ffcd3fe82b0

Possible return address

0x7ffcd3fe82b8

Since a, b, c are arguments to the called function, they are in the top of the stack followed by the local variables declared in order. The possible address where the return address for the function may be stored at or greater than 0x7ffcd3fe82b8

8.

a)  Garbage collection was first implemented in Lisp to avoid manual memory management. It was invented by John McCarthy

b) In languages like C which supports pointer, even if the name of the entity is not accessible and is out of scope, the entity can be accessed through pointers outside the scope. So, It should not be destroyed. But in other languages where there is no direct memory access, if the name of the entity is not accessible, entity can be destroyed

c) If the global variables are used as read-only variables like CONSTANTS, then It can be used in multi-threaded programs. It cannot be used in multi-threaded programs if there is read & write operations over it which may lead to race condition or race hazard.

d) A scanner need not always discard whitespace and comments. In languages like Python, whitespace indentation denotes a new block, so whitespace must be preserved during the analysis. And in some other languages, Comments will be needed to debug during program execution. So comments can also be preserved.

e) A stack is used to store data specific to a procedure. It will be created during the creation of a procedure and will be destroyed during the exiting of the procedure. So even if an entity having arbitrary lifetime is allocated in stack, it will be destroyed once the procedure execution is completed.