Pandora

Universidade Federal de Santa Catarina
Construção de Compiladores
Alunos: - Gabriel Luz
- Victor Feijó

Entrega I

Definição da Linguagem e Analise Lexica

Escolhemos a linguagem de programação Lua como base da definição de nossa gramática por se tratar de uma linguagem simples e moderna, e também foi desenvolvida no Brasil na PUC-RJ

Fizemos algumas mudanças na linguagem para simplificar e também deixar ela mais moderna.
As mudanças mais aparentes são as definições de variaveis agora é somente por def, definição de função é defn, inclusão de declaração arrow functions. Também removemos loops desnecessários para simplifcação da analise semantica. Agora, While é a única opção para loop da linguagem. Uma troca simples foi a troca sintática de como fazer comentarios. Originalmente o comentário de linha é --, foi trocado para os padrões definidos em C // para comentario em linha e /* */ para comentarios multi-linha

Definição de Variaveis

```
def foo = 'String'
def foo, bar = 'String', 20
def somaMult = (20 + 20) * 2
```

Definição de Funções

```
defn sum(a, b)
  return a + b
end
```

Definição de função atribuida a uma variável

```
def new_sum = function(a, b)
  return a + b
end
```

Definição com arrow function

```
def sub = (a, b) => return a - b end
```

EBNF da Linguagem

```
grammar Pandora;

chunk
    : block EOF
    ;

block
    : stat* retstat?
    ;

stat
    : ';'
    | functioncall
    | 'break'
    | 'while' exp block 'end'
    | 'if' exp 'then' block ('elseif' exp 'then' block)* ('else' block)? 'end'
    | 'def'? varlist ('=' explist)?
    | 'defn' funcname funcbody
    ;

retstat
    : 'return' explist? ';'?
    ;

funcname
    : NAME (':' NAME)?
    ;

varlist
    : var (',' var)*
    ;

namelist
    : NAME (',' NAME)*
    ;

explist
    : exp (',' exp)*
    ;

exp
    : 'nil' | 'false' | 'true'
    | number
    | string
    | '...'
    | functiondef
    | prefixexp
    | tableconstructor
    | <assoc=right> exp operatorPower exp
    | operatorUnary exp
    | exp operatorMulDivMod exp
    | exp operatorAddSub exp
    | <assoc=right> exp operatorStrcat exp
    | exp operatorComparison exp
    | exp operatorAnd exp
    | exp operatorOr exp
    | exp operatorBitwise exp
    ;

prefixexp
    : varOrExp nameAndArgs*
    ;

functioncall
    : varOrExp nameAndArgs+
    ;

varOrExp
    : var | '(' exp ')'
    ;

var
    : NAME varSuffix*
    ;
```

```
varSuffix
    : nameAndArgs* ('[' exp ']' | '.' NAME)
    ;

nameAndArgs
    : (':' NAME)? args
    ;

args
    : '(' explist? ')' | tableconstructor | string
    ;

functiondef
    : 'function' funcbody
    | '(' parlist? ')' '=>' block 'end'
    ;

funcbody
    : '(' parlist? ')' block 'end'
    ;

parlist
    : namelist (',' '...')? | '...'
    ;

tableconstructor
    : '{' fieldlist? '}'
    ;

fieldlist
    : field (fieldsep field)* fieldsep?
    ;

field
    : '[' exp ']' '=' exp | NAME '=' exp | exp
    ;

fieldsep
    : ',' | ';'
    ;

operatorOr
  : 'or';

operatorAnd
  : 'and';

operatorComparison
  : '<' | '>' | '<=' | '>=' | '~=' | '==';

operatorStrcat
  : '..';

operatorAddSub
  : '+' | '-';

operatorMulDivMod
  : '*' | '/' | '%';

operatorBitwise
  : '&' | '|' | '~' | '<<' | '>>';

operatorUnary
    : 'not' | '#' | '-' | '~';

operatorPower
    : '^';

number
    : INT | HEX | FLOAT | HEX_FLOAT
    ;

string
```

```
    : NORMALSTRING | CHARSTRING | LONGSTRING
    ;

// LEXER

NAME
    : [a-zA-Z_][a-zA-Z_0-9]*
    ;

NORMALSTRING
    : '"' ( EscapeSequence | ~('\\'|'"') )* '"'
    ;

CHARSTRING
    : '\'' ( EscapeSequence | ~('\''|'\\') )* '\''
    ;

LONGSTRING
    : '[' NESTED_STR ']'
    ;

fragment
NESTED_STR
    : '=' NESTED_STR '='
    | '[' .*? ']'
    ;

INT
    : Digit+
    ;

HEX
    : '0' [xX] HexDigit+
    ;

FLOAT
    : Digit+ '.' Digit* ExponentPart?
    | '.' Digit+ ExponentPart?
    | Digit+ ExponentPart
    ;

HEX_FLOAT
    : '0' [xX] HexDigit+ '.' HexDigit* HexExponentPart?
    | '0' [xX] '.' HexDigit+ HexExponentPart?
    | '0' [xX] HexDigit+ HexExponentPart
    ;

fragment
ExponentPart
    : [eE] [+-]? Digit+
    ;

fragment
HexExponentPart
    : [pP] [+-]? Digit+
    ;

fragment
EscapeSequence
    : '\\' [abfnrtvz"'\\]
    | '\\' '\r'? '\n'
    | DecimalEscape
    | HexEscape
    | UtfEscape
    ;

fragment
DecimalEscape
    : '\\' Digit
    | '\\' Digit Digit
    | '\\' [0-2] Digit Digit
    ;

fragment
```

```
HexEscape
    : '\\' 'x' HexDigit HexDigit
    ;

fragment
UtfEscape
    : '\\' 'u{' HexDigit+ '}'
    ;

fragment
Digit
    : [0-9]
    ;

fragment
HexDigit
    : [0-9a-fA-F]
    ;

COMMENT
    : '/*' .*? '*/' -> skip
    ;

LINE_COMMENT
    : '//' ~[\r\n]* -> skip
    ;

WS
    : [ \t\u000C\r\n]+ -> skip
    ;

SHEBANG
    : '#' '!' ~('\n'|'\r')* -> channel(HIDDEN)
    ;
```

Geração de Tokens

Exemplo 1

```
defn sum(a, b)
  return a + b
end
```

```
 [@0,0:3='defn',<'defn'>,1:0]
[@1,5:7='sum',<NAME>,1:5]
[@2,8:8='(',<'('>,1:8]
[@3,9:9='a',<NAME>,1:9]
[@4,10:10=',',<','>,1:10]
[@5,12:12='b',<NAME>,1:12]
[@6,13:13=')',<')'>,1:13]
[@7,17:22='return',<'return'>,2:2]
[@8,24:24='a',<NAME>,2:9]
[@9,26:26='+',<'+'>,2:11]
[@10,28:28='b',<NAME>,2:13]
[@11,30:32='end',<'end'>,3:0]
[@12,34:33='<EOF>',<EOF>,4:0]
```

Exemplo 2

```
def sub = (a, b) => return a - b end
```

```
 [@0,0:2='def',<'def'>,1:0]
[@1,4:6='sub',<NAME>,1:4]
[@2,8:8='=',<'='>,1:8]
[@3,10:10='(',<'('>,1:10]
[@4,11:11='a',<NAME>,1:11]
[@5,12:12=',',<','>,1:12]
[@6,14:14='b',<NAME>,1:14]
[@7,15:15=')',<')'>,1:15]
[@8,17:18='=>',<'=>'>,1:17]
[@9,20:25='return',<'return'>,1:20]
[@10,27:27='a',<NAME>,1:27]
[@11,29:29='-',<'-'>,1:29]
[@12,31:31='b',<NAME>,1:31]
[@13,33:35='end',<'end'>,1:33]
[@14,37:36='<EOF>',<EOF>,2:0]
```

Exemplos de erros léxicos

Exemplo 1

```
 defn hello

 [@0,0:3='defn',<'defn'>,1:0]
[@1,5:9='hello',<NAME>,1:5]
[@2,11:10='<EOF>',<EOF>,2:0]
line 2:0 mismatched input '<EOF>' expecting '('
```

```
def sub = (a, b) => return a - b
def foo = 'String'
def foo, bar = 'String', 20
def somaMult = (20 + 20) * 2

[@0,0:2='def',<'def'>,1:0]
[@1,4:6='sub',<NAME>,1:4]
[@2,8:8='=',<'='>,1:8]
[@3,10:10='(',<'('>,1:10]
[@4,11:11='a',<NAME>,1:11]
[@5,12:12=',',<','>,1:12]
[@6,14:14='b',<NAME>,1:14]
[@7,15:15=')',<')'>,1:15]
[@8,17:18='=>',<'=>'>,1:17]
[@9,20:25='return',<'return'>,1:20]
[@10,27:27='a',<NAME>,1:27]
[@11,29:29='-',<'-'>,1:29]
[@12,31:31='b',<NAME>,1:31]
[@13,33:35='def',<'def'>,2:0]
[@14,37:39='foo',<NAME>,2:4]
[@15,41:41='=',<'='>,2:8]
[@16,43:50=''String'',<CHARSTRING>,2:10]
[@17,52:54='def',<'def'>,3:0]
[@18,56:58='foo',<NAME>,3:4]
[@19,59:59=',',<','>,3:7]
[@20,61:63='bar',<NAME>,3:9]
[@21,65:65='=',<'='>,3:13]
[@22,67:74=''String'',<CHARSTRING>,3:15]
[@23,75:75=',',<','>,3:23]
[@24,77:78='20',<INT>,3:25]
[@25,80:82='def',<'def'>,4:0]
[@26,84:91='somaMult',<NAME>,4:4]
[@27,93:93='=',<'='>,4:13]
[@28,95:95='(',<'('>,4:15]
[@29,96:97='20',<INT>,4:16]
[@30,99:99='+',<'+'>,4:19]
[@31,101:102='20',<INT>,4:21]
[@32,103:103=')',<')'>,4:23]
[@33,105:105='*',<'*'>,4:25]
[@34,107:107='2',<INT>,4:27]
[@35,109:108='<EOF>',<EOF>,5:0]
line 2:0 missing 'end' at 'def'
```