Mohammed Umaruddin Assignment 2:

Section 1: Transformed grammar LL(1)

ARRAYSIZE has a first set conflict

EXPR has a first set conflict

FACTOR has a first set conflict

IDNEST has a first set conflict

STATEMENT has a first set conflict

There were various additional ambiguities while I was processing the grammar. Like aParms, intersecting first and follow sets.

Grammar fixed to LL(1):

START -> prog

prog -> rept-prog0

rept-prog0 -> classOrImplOrFunc rept-prog0

| EPSILON

classOrImplOrFunc -> funcDef | classDecl | implDef

classDecl -> class id opt-classDecl2 { rept-classDecl4 } ;

opt-classDecl2 -> isa id rept-opt-classDecl22

| EPSILON

rept-opt-classDecl22 -> , id rept-opt-classDecl22

| EPSILON

rept-classDecl4 -> visibility memberDecl rept-classDecl4

| EPSILON

implDef -> implementation id { rept-implDef3 }

rept-implDef3 -> funcDef rept-implDef3

| EPSILON

funcDef -> funcHead funcBody

funcHead -> function id ( fParams ) arrow returnType

| constructor ( fParams )

funcBody -> { rept-funcBody1 }

rept-funcBody1 -> localVarDeclOrStat rept-funcBody1

| EPSILON

localVarDeclOrStat -> statement | localVarDecl

arithExpr -> term rightrec-arithExpr

rightrec-arithExpr -> addOp term rightrec-arithExpr

| EPSILON

term -> factor rightrec-term

rightrec-term -> multOp factor rightrec-term

| EPSILON

factor -> ( arithExpr )

| floatLit

| intLit

| not factor

| idOrSelf factor2 VarOrFunc

| sign factor

factor2 -> ( aParams ) | rept-idnest10

VarOrFunc -> idnest VarOrFunc | EPSILON

arraySize -> [ arraySizeTail

arraySizeTail -> intNum ] | ]

type -> float | id | int

returnType -> void | type

memberDecl -> attributeDecl | funcDecl

funcDecl -> funcHead ;

attributeDecl -> attribute varDecl

varDecl -> id : type rept-varDecl3 ;

rept-varDecl3 -> arraySize rept-varDecl3

| EPSILON

expr -> arithExpr relExprRest

relExprRest -> relOp arithExpr | EPSILON

fParams -> id : type rept-fParams3 rept-fParams4

| EPSILON

rept-fParams3 -> arraySize rept-fParams3

| EPSILON

rept-fParams4 -> fParamsTail rept-fParams4

| EPSILON

fParamsTail -> , id : type rept-fParamsTail4

rept-fParamsTail4 -> arraySize rept-fParamsTail4

| EPSILON

aParams -> expr rept-aParams1 | EPSILON

rept-aParams1 -> aParamsTail rept-aParams1

| EPSILON

aParamsTail -> , expr

idOrSelf -> id | self

idnest -> . id idnest2

idnest2 -> ( aParams ) | rept-idnest10

rept-idnest10 -> indice rept-idnest10

| EPSILON

indice -> [ arithExpr ]

variable -> idOrSelf Variable2

Variable2 -> ( aParams ) idnest

| rept-idnest10 rept-variable

rept-variable -> varidnest rept-variable

| EPSILON

varidnest -> . id varidnest2

varidnest2 -> ( aParams ) varidnest

| rept-idnest10

localVarDecl -> local varDecl

relExpr -> arithExpr relOp arithExpr

assignOp -> :=

relOp -> < | <= | <> | == | > | >=

multOp -> \* | / | and

addOp -> + | - | or

sign -> + | -

statBlock -> { rept-statBlock1 }

| statement

| EPSILON

rept-statBlock1 -> statement rept-statBlock1

| EPSILON

statement -> if ( relExpr ) then statBlock else statBlock ;

| read ( variable ) ;

| return ( expr ) ;

| while ( relExpr ) statBlock ;

| write ( expr ) ;

| statement2

statement2 -> self statementidnest . statement2

| id stateVar

statementidnest -> ( aParams ) | rept-idnest10

stateVar -> ( aParams ) finishfunc

| rept-idnest10 finishvar

finishvar -> assignOp expr ; | stateFuncOrVar

finishfunc -> ; | stateFuncOrVar

stateFuncOrVar -> . statenest2

statenest2 -> id statementnest3

| self statementidnest stateFuncOrVar

statementnest3 -> ( aParams ) finishfunc

| rept-idnest10 finishvar

visibility -> private | public

Section 3:

I used the recursive descent approach. I made an unordered\_map for first and follow sets, where the name of the function is the key and it holds array of arrays of tokentype(the first and follow sets terminals). With the standard approach of recursive descent, the parser goes from function to function, searches the first and follow sets and returns the result which is true or false.

Section 4:

I have used the grammar tool provided by the professor. Although useful, it needed some changes in the input file to first run. I have also used atoCC to generate the first and follow sets for the grammar.