GROUP 39

AKANKSHYA MISHRA 2016A7PS0026P
NARAPAREDDY BHAVANA 2016A7PS0034P
KARABEE BATTA 2016A7PS0052P
AASTHA KATARIA 2016A7PS0062P

SEMANTIC RULES

MAIN

Rule 0: program --> otherFunctions mainFunction	1. program.node = new Node(TAG_PROGRAM, NULL, new Node(TAG_FUN_LIST, otherFunctions.node), mainFunction.node)
Rule 1: mainFunction --> TK_MAIN stmts TK_END	1. mainFunction.node=newNode(TAG_ MAIN, NULL, stmts.node)
Rule 2: otherFunctions -->function $otherFunctions_1$	1. otherFunctions.node = ChildrenList(function.node, $otherFunctions_1$.node)
Rule 3: otherFunctions --> eps	1. otherFunctions.node=NULL

FUNCTION

Rule 4: function --> TK_FUNID input_par output_par TK_SEM stmts TK_END	1. function.node=new Node(TAG_FUNCTION, LeafNode(TK_FUNID), new Node(TAG_INPUT_PARS, NULL, input_par.node), new Node(TAG_OUTPUT_PARS, NULL, output_par.node), stmts.node)
Rule 5: input_par --> TK_INPUT TK_PARAMETER TK_LIST TK_SQL parameter_list TK_SQR	1. input_par.node=parameter_list.node
Rule 6: output_par --> TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL parameter_list TK_SQR	1. output_par.node=parameter_list.node

Rule 7: output_par --> eps	1. output_par.node = NULL
Rule 8: parameter_list -->dataType TK_ID remaining_list	1. parameter_list.node = ChildrenList(dataType.node, ChildrenList(LeafNode(TK_ID), remaining_list.node))
Rule 14: remaining_list --> TK_COMMA parameter_list	1. remaining_list.node = parameter_list.node
Rule 15: remaining_list --> eps	1. remaining_list.node = NULL

DATATYPE

Rule 9: dataType -->primitiveDatatype	1. dataType.node = primitiveDatatype.node
Rule 10: dataType -->constructedDatatype	1. dataType.node = constructedDatatype.node
Rule 11: primitiveDatatype --> TK_INT	1. primitiveDatatype.node = Leafnode(TK_INT)
Rule 12: primitiveDatatype --> TK_REAL	1. primitiveDatatype.node = Leafnode(TK_REAL)
Rule 13: constructedDatatype --> TK_RECORD TK_RECORDID	1. constructedDatatype.node = LeafNode(TK_RECORDID)

STMTS

Rule 16: stmts -->typeDefinitions declarations otherStmts returnStmt	1. stmts.node = new ChildrenList(new Node(TAG_TYPEDEFS, NULL, typeDefinitions.node), new Node(TAG_DECLARES, NULL, declarations.node), new Node(TAG_OTHERSTMTS, NULL, otherStmts.node), new Node(TAG_RETURNSTMT, NULL, returnStmt.node))

TYPE DEFINITIONS

Rule 17: typeDefinitions -->typeDefinition $typeDefinitions_1$	1. typeDefinitions.node = ChildrenList(typeDefinition.node, $typeDefinitions_1$.node)
Rule 18: typeDefinitions --> eps	1. typeDefinitions.node = NULL
Rule 19: typeDefinition --> TK_RECORD TK_RECORDID fieldDefinitions TK_ENDRECORD TK_SEM	1. typeDefinition.node = new Node(TAG_TYPEDEF, LeafNode(TK_RECORDID), fieldDefinitions.node)
Rule 20: fieldDefinitions -->fieldDefinition fieldDefinition moreFields	1. fieldDefinitions.node = ChildrenList(fieldDefinition.node, ChildrenList(fieldDefinition.node, moreFields.node))
Rule 21: fieldDefinition --> TK_TYPE primitiveDatatype TK_COLON TK_FIELDID TK_SEM	1. fieldDefinition.node = new Node(TAG_FIELDDEF, LeafNode(FIELDID), primitiveDataType.node)
Rule 22: moreFields -->fieldDefinition $moreFields_1$	1. moreFields.node = ChildrenList(fieldDefinition.node, $moreFields_1$.node)
Rule 23: moreFields --> eps	1. moreFields.node=NULL

DECLARATIONS

Rule 24: declarations -->declaration $declarations_1$	1. declarations.node = ChildrenList(declaration.node, $declarations_1$.node)
Rule 25: declarations --> eps	1. declarations.node = NULL
Rule 26: declaration --> TK_TYPE dataType TK_COLON TK_ID global_or_not TK_SEM	1. declaration.node=new Node(TAG_DECLARE, LeafNode(TK_ID), dataType.node, global_or_not.node)
Rule 27: global_or_not --> TK_COLON TK_GLOBAL	1. global_or_not.node=LeafNode(TK_GLOBAL)
Rule 28: global_or_not --> eps	1. global_or_not.node = NULL

OTHERSTMTS

Rule 29: otherStmts -->stmt $otherStmts_1$	1. otherStmts.node = ChildrenList(stmt.node, $otherStmts_1$.node)
Rule 30: otherStmts --> eps	1. otherStmts.node = NULL

RETURN STATEMENT

Rule 81: returnStmt --> TK_RETURN optionalReturn TK_SEM	1. returnStmt.node=optionalReturn.node
Rule 82: optionalReturn --> TK_SQL idList TK_SQR	1. optionalReturn.node=idList.node
Rule 83: optionalReturn --> eps	1. optionalReturn.node = NULL
Rule 84: idList --> TK_ID more_ids	1. idList.node = ChildrenList(LeafNode(TK_ID), more_ids.node)
Rule 85: more_ids --> TK_COMMA idList	1. more_ids.node=idList.node

Rule 86: more_ids --> eps	1. more_ids.node =NULL

STMT

Rule 31: stmt -->assignmentStmt	1. stmt.node = assignmentStmt.node
Rule 32: stmt -->iterativeStmt	1. stmt.node = iterativeStmt.node
Rule 33: stmt -->conditionalStmt	1. stmt.node= conditionalStmt.node
Rule 34: stmt -->ioStmt	1. stmt.node= ioStmt.node
Rule 35: stmt -->funCallStmt	1. stmt.node = funCallStmt.node

ASSIGNMENT STATEMENT

Rule 36: assignmentStmt -->singleOrRecId TK_ASSIGNOP arithmeticExpression TK_SEM	1. assignmentStmt.node = new Node(TAG_ASSIGNMENT_STMT, LeafNode(TK_ASSIGNOP), singleOrRecId.node, arithmeticExpression.node)
Rule 37: singleOrRecId --> TK_ID new_24	1. singleOrRecId.node = ChildrenList(LeafNode(TK_ID), new_24.node)
Rule 38: new_24 --> eps	1. new_24.node = NULL
Rule 39: new_24 --> TK_DOT TK_FIELDID	1. new_24.node = LeafNode(TK_FIELDID)

ITERATIVE STATEMENT

Rule 44: iterativeStmt --> TK_WHILE TK_OP booleanExpression TK_CL stmt otherStmts TK_ENDWHILE	1. iterativeStmt.node = new Node(TAG_ITERATIVE_STMT, NULL, booleanExpression.node, stmt.node, otherStmts.node)

CONDITIONAL STATEMENT

Rule 45: conditionalStmt --> TK_IF TK_OP booleanExpression TK_CL TK_THEN stmt otherStmts elsePart	1. conditionalStmt.node = newNode(TAG_COND_STMT, NULL, booleanExpression.node, stmt, otherStmts, elsePart.node)
Rule 46: elsePart --> TK_ELSE stmt otherStmts TK_ENDIF	1. elsePart.node=ChildrenList(stmt.node, otherStmts.node)
Rule 47: elsePart --> TK_ENDIF	1. elsePart.node = NULL

IO STATEMENT

Rule 48: ioStmt --> TK_READ TK_OP singleOrRecId TK_CL TK_SEM	1. ioStmt.node = new Node(TAG_READ, NULL, singleOrRecId.node)
Rule 49: ioStmt --> TK_WRITE TK_OP allVar TK_CL TK_SEM	1. ioStmt.node = new Node(TAG_WRITE, NULL, allVar.node)

FUNCTION CALL STATEMENT

Rule 40: funCallStmt -->outputParameters TK_CALL TK_FUNID TK_WITH TK_PARAMETERS inputParameters TK_SEM	1. funCallStmt.node = new Node(FUN_CALL_STMT, LeafNode(TK_FUNID), new Node(TAG_OUTPUT_ARGS, NULL, outputParameters.node), new Node(TAG_INPUT_ARGS, NULL,

	inputParameters.node))
Rule 41: outputParameters --> TK_SQL idList TK_SQR TK_ASSIGNOP	1. outputParameters.node = idList.node
Rule 42: outputParameters --> eps	1. outputParameters.node=NULL
Rule 43: inputParameters --> TK_SQL idList TK_SQR	1. inputParameters.node = idList.node

ARITHMETIC EXPRESSION

Rule 50: allVar --> TK_NUM	1. allVar.node = LeafNode(TK_NUM)
Rule 51: allVar --> TK_RNUM	1. allVar.node = LeafNode(TK_RNUM)
Rule 52: allVar --> TK_ID temp	1. allVar.node = ChildrenList(LeafNode(TK_ID), temp.node)
Rule 53: arithmeticExpression -->term expPrime	1. arithmeticExpression.node = expPrime.node 2. expPrime.inh = term.node
Rule 54: expPrime→ lowPrecedenceOperators term $expPrime_1$	1. $expPrime_1$.inh = new Node(TAG_ARITHMETIC_EXP RESSION, lowPrecedenceOperators.node, expPrime.inh, term.node) 2. expPrime.node = $expPrime_1$.node
Rule 55: expPrime --> eps	1. expPrime.node = expPrime.inh
Rule 56: term -->factor termPrime	1. term.node = termPrime.node 2. termPrime.inh = factor.node
Rule 57: termPrime→ highPrecedenceOperators factor $termPrime_1$	1. $termPrime_1$.inh = new Node(TAG_ARITHMETIC_EXPRESSION, highPrecedenceOperators.node, termPrime.inh, factor.node) 2. termPrime.node = $termPrime_1$.node
Rule 58: termPrime --> eps	1. termPrime.nd = termPrime.inh

Rule 59: factor --> TK_OP arithmeticExpression TK_CL	1. factor.node = arithmeticExpression.node
Rule 60: factor -->allVar	1. factor.node = allVar.node
Rule 61: highPrecedenceOperators --> TK_MUL	1. highPrecedenceOperators.node = LeafNode(TK_MUL)
Rule 62: highPrecedenceOperators --> TK_DIV	1. highPrecedenceOperators.node = LeafNode(TK_DIV)
Rule 63: lowPrecedenceOperators --> TK_PLUS	1. lowPrecedenceOperators.node = LeafNode(TK_PLUS)
Rule 64: lowPrecedenceOperators --> TK_MINUS	1. lowPrecedenceOperators.node = LeafNode(TK_MINUS)
Rule 65: temp --> eps	1. temp.node = NULL
Rule 66: temp --> TK_DOT TK_FIELDID	1. temp.node = LeafNode(TK_FIELDID)

BOOLEAN EXPRESSION

Rule 67: booleanExpression --> TK_OP $booleanExpression_1$ TK_CL logicalOp TK_OP $booleanExpression_2$ TK_CL	1. booleanExpression.node = new Node(TAG_BOOLEAN_EXPRESSION, logicalOp.node, $booleanExpression_1$.node, $booleanExpression_2$.node)
Rule 68: booleanExpression -->var_1 relationalOp var_2	1. booleanExpression.node = new Node(TAG_BOOLEAN_EXPRESSION, relationalOp.node, var_1.node, var_2.node)
Rule 69: booleanExpression --> TK_NOT TK_OP $booleanExpression_1$ TK_CL	1. booleanExpression.node = new Node(TAG_BOOLEAN_EXPRESSION, leafNode(TK_NOT), $booleanExpression_1$.node)
Rule 70: var --> TK_ID	1. var.node = LeafNode(TK_ID)
Rule 71: var --> TK_NUM	1. var.node = LeafNode(TK_NUM)

Rule 72: var --> TK_RNUM	1. var.node = LeafNode(TK_RNUM)
Rule 73: logicalOp --> TK_AND	1. logicalOp.node = LeafNode(TK_AND)
Rule 74: logicalOp --> TK_OR	1. logicalOp.node = LeafNode(TK_OR)
Rule 75: relationalOp --> TK_LT	1. relationalOp.node = LeafNode(TK_LT)
Rule 76: relationalOp --> TK_LE	1. relationalOp.node = LeafNode(TK_LE)
Rule 77: relationalOp --> TK_EQ	1. relationalOp.node = LeafNode(TK_EQ)
Rule 78: relationalOp --> TK_GT	1. relationalOp.node = LeafNode(TK_GT)
Rule 79: relationalOp --> TK_GE	1. relationalOp.node = LeafNode(TK_GE)
Rule 80: relationalOp --> TK_NE	1. relationalOp.node = LeafNode(TK_NE)