Semantic Rules for Abstract Syntax Tree Generation

Submitted By
Naveen Venkat
2015A7PS0078P

Corresponding to every grammar rule (numbered), the semantic rules have been written in sequence each within curly braces. A C-like pseudo code is followed to represent the corresponding semantic rules. Non terminal nodes are within angular brackets and terminal nodes are in capital letters.

Attribute descriptions are as follows:

- carry : used as a synthesized attribute
- inh : used as an inherited attribute
- addr : denotes the address of the corresponding parse tree node
- link : used to link the node with a sibling node (as in the case of a linked list)

Each of the above four attributes can be viewed as pointers pointing to nodes.

Semantic rules are as follows:

1. <mainFunction> ===> MAIN SQO SQC <stmtsAndFunctionDefs> END

   {free(MAIN)}, {free(SQO)}, {free(SQC)}, {free(END)}, {<mainFunction> = newnode("MAINFUNCTION", <stmtsAndFunctionDefs>.addr)}

2. <stmtsAndFunctionDefs> ===> <stmtOrFunctionDef> <other1>

   {<stmtOrFunctionDef>.carry.link = <other1>.carry}, {<stmtsAndFunctionDefs> = newnode("SAFD", <stmtOrFunctionDef>.carry)}, {free(<other1>)}, {free(<stmtsAndFunctionDefs>)}

3. <other1> ===> _epsilon_

   {<other1>.carry=NULL}, {free(_epsilon_)}

4. <other1> ===> <stmtOrFunctionDef> <other1>

   {<other1>_1.carry = <stmtOrFunctionDef>.carry}, {<stmtOrFunctionDef>.carry.link = <other1>_2.carry}, {free(<other1>_2)}, {free(<stmtOrFunctionDef>)}

5. <stmtOrFunctionDef> ===> <stmt>

   {<stmtOrFunctionDef>.carry = <stmt>.carry}, {free(<stmt>)}

6. <stmtOrFunctionDef> ===> <functionDef>

   {<stmtOrFunctionDef>.carry = <functionDef>.addr}

7. <stmt> ===> <declarationStmt>

{<stmt>.carry = <declarationStmt>.addr}

8. <stmt> ===> <assignmentStmt_type1>

{<stmt>.carry = <assignmentStmt_type1>.addr}

9. <stmt> ===> <assignmentStmt_type2>

{<stmt>.carry = <assignmentStmt_type2>.addr}

10. <stmt> ===> <ifStmt>

{<stmt>.carry = <ifStmt>.addr}

11. <stmt> ===> <ioStmt>

{<stmt>.carry = <ioStmt>.addr}

12. <stmt> ===> <funCallStmt> SEMICOLON

{<stmt>.carry = <funCallStmt>.addr}, {free(SEMICOLON)}

13. <functionDef> ===> FUNCTION SQO <parameter_list> SQC ASSIGNOP FUNID SQO <parameter_list> SQC <stmtsAndFunctionDefs> END SEMICOLON

{free(FUNCTION)}, {free(SQO_1)}, {free(SQC_1)}, {free(ASSIGNOP)}, {free(SQO_2)}, {free(SQC_2)}, {free(END)}, {free(SEMICOLON)}, {<functionDef> = newnode(<parameter_list>_1.addr, FUNID.addr, <parameter_list>_2.addr, <stmtsAndFunctionDefs>.addr)}

14. <parameter_list> ===> <type> ID <remainingList>

{<type>.link = ID.addr}, {ID.link = <remainingList>.carry}, {free(<remainingList>)}, {<parameter_list> = newnode("PL", <type>.addr)}

15. <remainingList> ===> COMMA <type> ID <remainingList>

{free(COMMA)}, {free(ID)}, {<type>.link = ID.addr}, {ID.link = <remainingList>_2.carry}, {<remainingList>_1.carry = <type>.addr}, {free(<remainingList>_2)}

16. <remainingList> ===> _epsilon_

{<remainingList>.carry = NULL}, {free(_epsilon_)}

17. <type> ===> INT

{<type>.carry = INT.addr}

18. <type> ===> REAL

{<type>.carry = REAL.addr}

19. <type> ===> STRING

{<type>.carry = STRING.addr}

20. <type> ===> MATRIX

{<type>.carry = MATRIX.addr}

21. <declarationStmt> ===> <type> <var_list> SEMICOLON

{free(SEMICOLON)}, {<declarationStmt> = newnode("DEC", <type>.carry, <var_list>.carry)}, {free(<var_list>)}, {free(<type>)}

22. <var_list> ===> ID <more_ids>

{ID.link = <more_ids>.carry}, {<var_list>.carry = ID.addr}, {free(<more_ids>)}

23. <more_ids> ===> COMMA ID <more_ids>

{ID.link = <more_ids>.carry}, {<more_ids>.carry = ID.addr}, {free(COMMA)}, {free(<more_ids>)}

24. <more_ids> ===> _epsilon_

{<more_ids>.carry = NULL}, {free(_epsilon_)}

25. <assignmentStmt_type1> ===> <leftHandSide_singleVar> ASSIGNOP <rightHandSide_type1> SEMICOLON

{free(SEMICOLON)}, {free(ASSIGNOP)}, {<assignmentStmt_type1> = newnode("=1", <leftHandSide_singleVar>.carry, <rightHandSide_type1>.carry)}, {free(<leftHandSide_singleVar>)}, {free(<rightHandSide_type1>)}

26. <assignmentStmt_type2> ===> <leftHandSide_listVar> ASSIGNOP <rightHandSide_type2> SEMICOLON

{free(SEMICOLON)}, {free(ASSIGNOP)}, {<assignmentStmt_type2> = newnode("=2", <leftHandSide_listVar>.carry, <rightHandSide_type2>.carry)}, {free(<leftHandSide_listVar>)}, {free(<rightHandSide_type2>)}

27. <leftHandSide_singleVar> ===> ID

{<leftHandSide_singleVar>.carry = ID.addr}

28. <leftHandSide_listVar> ===> SQO <var_list> SQC

{free(SQO)}, {free(SQC)}, {<leftHandSide_listVar>.carry = newnode(<var_list>.carry)}, {free(<var_list>)}

29. <rightHandSide_type1> ===> <arithmeticExpression>

{<rightHandSide_type1>.carry = <arithmeticExpression>.carry}, {free(<arithmeticExpression>)}

30. <rightHandSide_type1> ===> <sizeExpression>

{<rightHandSide_type1>.carry = <sizeExpression>.addr}

31. <rightHandSide_type1> ===> <funCallStmt>

{<rightHandSide_type1>.carry = <funCallStmt>.addr}

32. <rightHandSide_type2> ===> <sizeExpression>

{<rightHandSide_type2>.carry = <sizeExpression>.addr}

33. <rightHandSide_type2> ===> <funCallStmt>

{<rightHandSide_type2>.carry = <funCallStmt>.addr}

34. <sizeExpression> ===> SIZE ID

{free(SIZE)}, {<sizeExpression> = newnode("SIZE", ID.addr)}

35. <ifStmt> ===> IF OP <booleanExpression> CL <stmt> <otherStmts> <other2>

{free(IF)}, {free(OP)}, {free(CL)}, {<stmt>.link = <otherStmts>.carry}, {<ifStmt> = newnode("IF", <booleanExpression>.addr, <stmt>.carry, <other2>.carry)}, {free(<stmt>)}, {free(<other2>)}

36. <other2> ===> ELSE <stmt> <otherStmts> ENDIF SEMICOLON

{free(ELSE)}, {free(ENDIF)}, {free(SEMICOLON)}, {<stmt>.carry.link = <otherStmts>.carry}, {<other2>.carry = newnode("ELSE", <stmt>.carry)}, {free(<stmt>)}, {free(<otherStmts>)}

37. <other2> ===> ENDIF SEMICOLON

{<other2>.carry = NULL}, {free(ENDIF)}, {free(SEMICOLON)}

38. <otherStmts> ===> <stmt> <otherStmts>

{<otherStmts>_1.carry = <stmt>}, {<stmt>.link = <otherStmts>_2.carry}, {free(<otherStmts>_2)}

39. <otherStmts> ===> _epsilon_

{<otherStmts>.carry = NULL}

40. <ioStmt> ===> READ OP ID CL SEMICOLON

{free(READ)}, {free(OP)}, {free(CL)}, {free(SEMICOLON)}, {<ioStmt> = newnode("IO_READ", ID.addr)}

41. <ioStmt> ===> PRINT OP ID CL SEMICOLON

{free(PRINT)}, {free(OP)}, {free(CL)}, {free(SEMICOLON)}, {<ioStmt> = newnode("IO_PRINT", ID.addr)}

42. <funCallStmt> ===> FUNID OP <inputParameterList> CL

{free(OP)}, {free(CL)}, {<funCallStmt> = newnode("CALL", FUNID.addr, <inputParameterList>.addr)

43. <inputParameterList> ===> <var> <listVar>

{<var>.carry.link = <listVar>.carry}, {free(<var>)}, {free(<listVar>)}

44. <inputParameterList> ===> _epsilon_

{<inputParameterList>.carry = NULL}, {free(_epsilon_)}

45. <listVar> ===> _epsilon_

{<listVar>.carry = NULL}, {free(_epsilon_)}

46. <listVar> ===> COMMA <var> <listVar>

{free(COMMA)}, {<var>.carry.link = <listVar>_2.carry}, {free(<listVar>_2)}, {free(<var>)}

47. <arithmeticExpression> ===> <arithmeticTerm> <other3>

{<other3>.inh = <arithmeticTerm>.carry}, {<arithmeticTerm>.carry = <other3>.carry}, {free(<other3>)}

48. <other3> ===> <operator_lowPrecedence> <arithmeticTerm> <other3>

{<other3>_2.inh = <arithmeticTerm>.carry}, {<other3>_1.carry = newnode(<operator_lowPrecedence>.symbol, <other3>_1.inh, <arithmeticTerm>.carry)}, {free(<other3>_2)}, {free(<operator_lowPrecedence>)}, {free(<arithmeticTerm>)}

49. <other3> ===> _epsilon_

{<other3>.carry = <other3>.inh}, {free(_epsilon_)}

50. <arithmeticTerm> ===> <factor> <other4>

{<other4>.inh = <factor>.carry}, {<arithmeticTerm>.carry = <other4>.carry}, {free(<other4>)}

51. <other4> ===> <operator_highPrecedence> <factor> <other4>

{<other4>_2.inh = <factor>.carry}, {<other4>_1.carry = newnode(<operator_highPrecedence>.symbol, <other4>_1.inh, <factor>.carry)}, {free(<other4>_2)}, {free(<operator_highPrecedence>)}, {free(<factor>)}

52. <other4> ===> _epsilon_

{<other4>.carry = <other4>.inh}, {free(_epsilon_)}

53. <factor> ===> OP <arithmeticExpression> CL

{free(OP)}, {free(CL)}, {<factor>.carry = <arithmeticExpression>.carry}, {free(<arithmeticExpression>)}

54. <factor> ===> <var>

{<factor>.carry = <var>.carry}, {free(<var>)}

55. <operator_lowPrecedence> ===> PLUS

{<operator_lowPrecedence>.carry = PLUS.addr}

56. <operator_lowPrecedence> ===> MINUS

{<operator_lowPrecedence>.carry = MINUS.addr}

57. <operator_highPrecedence> ===> MUL

{<operator_highPrecedence>.carry = MUL.addr}

58. <operator_highPrecedence> ===> DIV

{<operator_highPrecedence>.carry = DIV.addr}

59. <booleanExpression> ===> OP <booleanExpression> CL <logicalOp> OP <booleanExpression> CL

{free(OP_1)}, {free(CL_1)}, {free(OP_2)}, {free(CL_2)}, {<booleanExpression>_1 = newnode(<logicalOp>.carry.symbol, <booleanExpression>_2, <booleanExpression>_3)}

60. <booleanExpression> ===> <constrainedVars> <relationalOp> <constrainedVars>

{<booleanExpression> = newnode(<relationalOp>.carry.symbol, <constrinedVars>_1.carry, <constrinedVars>_2.carry)}, {free(<constrainedVars>_1)}, {free(<constrainedVars>_2)}

61. <booleanExpression> ===> NOT OP <booleanExpression> CL

{free(NOT)}, {free(OP)}, {free(CL)}, {<booleanExpression>_1 = newnode("NOT", <booleanExpression>_2)}

62. <constrainedVars> ===> ID

{<constrinedVars>.carry = ID.addr}

63. <constrainedVars> ===> NUM

{<constrinedVars>.carry = NUM.addr}

64. <constrainedVars> ===> RNUM

{<constrinedVars>.carry = RNUM.addr}

65. <var> ===> ID <matrixElementExtension>

{ID.link = <matrixElementExtension>.carry}, {<var>.carry = ID}, {free(<matrixElementExtension>)}

66. <var> ===> NUM

{<var>.carry = NUM.addr}

67. <var> ===> RNUM

{<var>.carry = RNUM.addr}

68. <var> ===> STR

{<var>.carry = STR.addr}

69. <var> ===> <matrix>

{<var>.carry = <matrix>.addr}

70. <matrix> ===> SQO <rows> SQC

{free(SQO)}, {free(SQC)}, {<matrix> = newnode("MATRIX", <rows>.carry)}, {free(<rows>)}

71. <rows> ===> <row> <other5>

{<row>.link = <other5>.carry}, {<rows>.carry = <row>.addr}, {free(<other5>)}

72. <other5> ===> SEMICOLON <row> <other5>

{<row>.link = <other5>_2.carry}, {free(<other5>_2)}, {free(SEMICOLON)}, {<other5>_1.carry = <row>.addr}

73. <other5> ===> _epsilon_

{<other5>.carry = NULL}, {free(_epsilon_)}

74. <row> ===> NUM <remainingColElements>

{NUM.link = <remainingColElements>.carry}, {free(<remainingColElements>)}

75. <remainingColElements> ===> COMMA NUM <remainingColElements>

{free(COMMA)}, {<remainingColElements>_1.carry = NUM.addr}, {NUM.link = <remainingColElements>_2.carry}, {free(<remainingColElements>_2)}

76. <remainingColElements> ===> _epsilon_

{<remainingColElements>.carry = NULL}, {free(_epsilon_)}

77. <matrixElementExtension> ===> SQO NUM COMMA NUM SQC

{free(SQO)}, {free(COMMA)}, {free(SQC)}, {<matrixElemenetExtension>.carry = newnode("indices", NUM_1.addr, NUM_2.addr)}

78. <matrixElementExtension> ===> _epsilon_

{<matrixElementExtension>.carry = NULL}, {free(_epsilon_)}

79. <logicalOp> ===> AND

{<logicalOp>.carry = AND.addr}

80. <logicalOp> ===> OR

{<logicalOp>.carry = OR.addr}

81. <relationalOp> ===> LT

{<relationalOp>.carry = LT.addr}

82. <relationalOp> ===> LE

{<relationalOp>.carry = LE.addr}

83. <relationalOp> ===> EQ

{<relationalOp>.carry = EQ.addr}

84. <relationalOp> ===> GT

{<relationalOp>.carry = GT.addr}

85. <relationalOp> ===> GE

{<relationalOp>.carry = GE.addr}

86. <relationalOp> ===> NE

{<relationalOp>.carry = NE.addr}