Semantic Rules – AST Creation

Group 50

ANIRUDDHA MAHAJAN – 2017A7PS0145P
RAVINDRA SINGH SHEKHAWAT – 2017A7PS0146P
SHREYAS SRIKRISHNA – 2017A7PS0162P
CHETAN SHARMA – 2017A7PS0182P

NOTE: *All parse tree nodes on the right hand side of each rule are freed after execution of all semantic rules for that rule.*

1. <program> --> <moduleDeclarations><otherModules><driverModule><otherModules%>
    - <moduleDeclarations>.inh = NULL;
    - <otherModules>.inh = NULL;
    - <otherModules%>.inh = NULL;
    - <program>.syn = new ProgramNode(<moduleDeclarations>.syn, <otherModules>.syn, <driverModule>.syn, <otherModules%>.syn);

2. <moduleDeclarations> --> <moduleDeclaration><moduleDeclarations%>
    - <moduleDeclarations%>.inh = <moduleDeclaration>.syn;
    - insertAtBeginning(<moduleDeclarations>.inh,<moduleDeclarations%>.syn);
    - <moduleDeclarations>.syn = <moduleDeclarations%>.syn;

3. <moduleDeclarations> --> #
    - if(<moduleDeclarations>.inh == NULL)
      <moduleDeclarations>.syn = new NullNode();
    - else
      <moduleDeclarations>.syn = <moduleDeclarations>.inh;

4. <moduleDeclaration> --> DECLARE MODULE ID SEMICOL
    - <moduleDeclaration>.syn = new IdNode(ID.tkn);

5. <otherModules> --> <module><otherModules%>
    - <otherModules%>.inh = <module>.syn;
    - insertAtBeginning(<otherModules>.inh,<otherModules%>.syn);
    - <otherModules>.syn = <otherModules%>.syn;

6. <otherModules> --> #
    - If(<otherModules>.inh == NULL)
      <otherModules>.syn = new NullNode();
    - else
      <otherModules>.syn = <otherModules>.inh;

7. <driverModule> --> DRIVERDEF DRIVER PROGRAM DRIVERENDDEF <moduleDef>
    - <driverModule>.syn = <moduleDef>.syn;

8. <module> --> DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL <ret><moduleDef>
    - <module>.syn = new ModuleNode(new IdNode(ID.tkn), <input_plist>.syn, <ret>.syn, <moduleDef>.syn);

9. <ret> --> RETURNS SQBO <output_plist> SQBC SEMICOL
    - <ret>.syn = <output_plist>.syn;

10. <ret> --> #
    - <ret>.syn = new NullNode();

11. <input_plist> --> ID COLON <dataType><N1>
    - <N1>.inh = new InputParamNode(new IdNode(ID.tkn), <dataType>.syn);
    - <input_plist>.syn = <N1>.syn;

12. <N1> --> COMMA ID COLON <dataType><N1%>
    - <N1%>.inh = new InputParamNode(new IdNode(ID.tkn), <dataType>.syn);
    - if(<N1>.inh != NULL)
        insertAtBeginning(<N1>.inh, <N1%>.syn);
    - <N1>.syn = <N1%>.syn;

13. <N1> --> #
    - <N1>.syn = <N1>.inh;

14. <output_plist> --> ID COLON <type><N2>
    - <N2>.inh = new OutputParamNode(new IdNode(ID.tkn), <type>.syn);
    - <output_plist>.syn = <N2>.syn;

15. <N2> --> COMMA ID COLON <type><N2%>
    - <N2%>.inh = new OutputParamNode(new IdNode(ID.tkn),<type>.syn);
    - if(<N2>.inh != NULL)
        insertAtBeginning(<N2>.inh, <N2%>.syn);
    - <N2>.syn = <N2%>.syn;

16. <N2> --> #
    - <N2>.syn = <N2>.inh;

17. <dataType> --> ARRAY SQBO <range_arrays> SQBC OF <type>
    - <dataType>.syn = new ArrayTypeNode(<range_arrays>.syn, <type>.syn);

18. <dataType> --> INTEGER
    - <dataType>.syn = new TypeNode(INTEGER.tkn);

19. <dataType> --> REAL
    - <dataType>.syn = new TypeNode(REAL.tkn);

20. <dataType> --> BOOLEAN

- <dataType>.syn = new TypeNode(BOOLEAN.tkn);

21. <range_arrays> --> <Index> RANGEOP <Index%>
    - <range_arrays>.syn = new RangeNode(<Index>.syn, <Index%>.syn);

22. <type> --> INTEGER
    - <type>.syn = new TypeNode(INTEGER.tkn);

23. <type> --> REAL
    - <type>.syn = new TypeNode(REAL.tkn);

24. <type> --> BOOLEAN
    - <type>.syn = new TypeNode(BOOLEAN.tkn);

25. <moduleDef> --> START <statements> END
    - <statements>.inh = NULL;
    - <moduleDef>.syn = <statements>.syn

26. <statements> --> <statement><statements%>
    - <statements%>.inh = <statement>.syn;
    - If(<statements>.inh != NULL)
      insertAtBeginning(<statements>.inh, <statements%>.syn);
    - <statements>.syn = <statements%>.syn;

27. <statements> --> #
    - If(<statements>.inh == NULL)
      <statements>.syn = new NullNode();
    - Else
      <statements>.syn = <statements>.inh;

28. <statement> --> <ioStmt>
    - <statement>.syn = <ioStmt>.syn;

29. <statement> --> <simpleStmt>
    - <statement>.syn = <simpleStmt>.syn;

30. <statement> --> <declareStmt>
    - <statement>.syn = <declareStmt>.syn;

31. <statement> --> <conditionalStmt>
    - <statement>.syn = <conditionalStmt>.syn;

32. <statement> --> <iterative>
    - <statement>.syn = <iterative>.syn;

33. <iterative> --> FOR BO ID IN <range> BC START <statements> END
    - <statements>.inh = NULL;

- <iterative>.syn = new ForLoopNode(new IdNode(ID.tkn), <range>.syn, <statements>.syn);

34. <iterative> --> WHILE BO <arithOrBoolExpr> BC START <statements> END
    - <statements>.inh = NULL;
    - <iterative>.syn = new WhileLoopNode(<arithOrBoolExpr>.syn,<statements>.syn);

35. <ioStmt> --> GET_VALUE BO ID BC SEMICOL
    - <ioStmt>.syn = new InputIONode(new IdNode(ID.tkn));

36. <ioStmt> --> PRINT BO <var> BC SEMICOL
    - <ioStmt>.syn = new OutputIONode(<var>.syn);

37. <var> --> <var_id_num>
    - <var>.syn = <var_id_num>.syn;

38. <var> --> <boolConstt>
    - <var>.syn = <boolConstt>.syn;

39. <whichId> --> SQBO <Index> SQBC
    - <whichId>.syn = new ArrayIdNode(<whichId>.inh, <Index>.syn);

40. <whichId> --> #
    - <whichId>.syn = <whichID>.inh;

41. <simpleStmt> --> <assignmentStmt>
    - <simpleStmt>.syn = <assignmentStmt>.syn;

42. <simpleStmt> --> <moduleReuseStmt>
    - <simpleStmt>.syn = <moduleReuseStmt>.syn;

43. <assignmentStmt> --> ID <whichStmt>
    - <whichStmt>.inh = new IdNode(ID.tkn);
    - <assignmentStmt>.syn = <whichStmt>.syn;

44. <whichStmt> --> <lvalueIDStmt>
    - <lvalueIDStmt>.inh = <whichStmt>.inh;
    - <whichStmt>.syn = <lvalueIDStmt>.syn;

45. <whichStmt> --> <lvalueARRStmt>
    - <lvalueARRStmt>.inh = <whichStmt>.inh;
    - <whichStmt>.syn = <lvalueARRStmt>.syn;

46. <lvalueIDStmt> --> ASSIGNOP <expression> SEMICOL
    - <lvalueIDStmt>.syn = new AssignmentNode(<lvalueIDStmt>.inh,<expression>.syn);

47. <lvalueARRStmt> --> SQBO <Index> SQBC ASSIGNOP <expression> SEMICOL
    - <lvalueARRStmt>.syn = new AssignmentNode(new ArrayIdNode(<lvalueARRStmt>.inh, <Index>.syn), <expression>.syn);

48. <Index> --> NUM
    - <Index>.syn = new NumNode(NUM.tkn);

49. <Index> --> ID
    - <Index>.syn = new IdNode(ID.tkn);

50. <moduleReuseStmt> --> <optional> USE MODULE ID WITH PARAMETERS <idList> SEMICOL
    - <optional>.inh = NULL;
    - <moduleReuseStmt>.syn = new FunctionCallNode(<optional>.syn, new IdNode(ID.tkn), <idList>.syn);

51. <optional> --> SQBO <idList> SQBC ASSIGNOP
    - <optional>.syn = <idList>.syn;

52. <optional> --> #
    - If(<optional>.inh == NULL)
        <optional>.syn = new NullNode();
    - Else
        <optional>.syn = <optional>.inh;

53. <idList> --> ID <N3>
    - <N3>.inh = new IdNode(ID.tkn);

54. <N3> --> COMMA ID <N3%>
    - <N3%>.inh = new IdNode(ID.tkn);
    - insertAtBeginning(<N3>.inh, <N3%>.syn);
    - <N3>.syn = <N3%>.syn;

55. <N3> --> #
    - <N3>.syn = <N3>.inh;

56. <expression> --> <arithOrBoolExpr>
    - <expression>.syn = <arithOrBoolExpr>.syn;

57. <expression> --> <U>
    - <expression>.syn = <U>.syn;

58. <U> --> <op1> <new_NT>
    - addChildren(<op1>.syn, <new_NT>.syn, NULL);
    - <U>.syn = <op1>.syn;

59. <new_NT> --> BO <arithmeticExpr> BC
    - <new_NT>.syn = <arithmeticExpr>.syn;

60. <new_NT> --> <var_id_num>
    - <new_NT>.syn = <var_id_num>.syn;

61. <arithOrBoolExpr> --> <AnyTerm><N7>
    - <N7>.inh = <AnyTerm>.syn;
    - <arithOrBoolExpr>.syn = <N7>.syn;

62. <N7> --> <logicalOp><AnyTerm><N7%>
    - addChildren(<logicalOp>.syn, <N7>.inh, <AnyTerm>.syn);
    - <N7%>.inh = <logicalOp>.syn;
    - <N7>.syn = <N7%>.syn;

63. <N7> --> #
    - <N7>.syn = <N7>.inh;

64. <AnyTerm> --> <arithmeticExpr> <N8>
    - <N8>.inh = <arithmeticExpr>.syn;
    - <AnyTerm>.syn = <N8>.syn;

65. <AnyTerm> --> <boolConstt>
    - <AnyTerm>.syn = <boolConstt>.syn;

66. <boolConstt> --> TRUE
    - <boolConstt>.syn = new BoolNode(TRUE.tkn);

67. <boolConstt> --> FALSE
    - <boolConstt>.syn = new BoolNode(FALSE.tkn);

68. <N8> --> <relationalOp> <arithmeticExpr>
    - addChildren(<relationalOp>.syn, <N8>.inh, <arithmeticExpr>.syn);
    - <N8>.syn = <relationalOp>.syn

69. <N8> --> #
    - <N8>.syn = <N8>.inh;

70. <arithmeticExpr> --> <term><N4>
    - <N4>.inh = <term>.syn;
    - <arithmeticExpr>.syn = <N4>.syn;

71. <N4> --> <op1><term><N4%>
    - addChildren(<op1>.syn, <N4>.inh, <term>.syn);
    - <N4%>.inh = <op1>.syn;
    - <N4>.syn = <N4%>.syn;

72. <N4> --> #
    - <N4>.syn = <N4>.inh;

73. <term> --> <factor><N5>
    - <N5>.inh = <factor>.syn;
    - <term>.syn = <N5>.syn;

74. <N5> --> <op2><factor><N5%>
    - addChildren(<op2>.syn,<N5>.inh,<factor>.syn);
    - <N5%>.inh = <op2>.syn;
    - <N5>.syn = <N5%>.syn;

75. <N5> --> #
    - <N5>.syn = <N5>.inh;

76. <factor> --> BO <arithOrBoolExpr> BC
    - <factor>.syn = <arithOrBoolExpr>.syn;

77. <factor> --> <var_id_num>
    - <factor>.syn = <var_id_num>.syn;

78. <var_id_num> --> ID <whichId>
    - <whichId>.inh = new IdNode(ID.tkn);
    - <var_id_num>.syn = <whichId>.syn;

79. <var_id_num> --> NUM
    - <var_id_num>.syn = new NumNode(NUM.tkn);

80. <var_id_num> --> RNUM
    - <var_id_num>.syn = new NumNode(RNUM.tkn);

81. <op1> --> PLUS
    - <op1>.syn = new OpNode(PLUS.tkn);

82. <op1> --> MINUS
    - <op1>.syn = new OpNode(MINUS.tkn);

83. <op2> --> MUL
    - <op1>.syn = new OpNode(MUL.tkn);

84. <op2> --> DIV
    - <op1>.syn = new OpNode(DIV.tkn);

85. <logicalOp> --> AND
    - <logicalOp>.syn = new OpNode(AND.tkn);

86. <logicalOp> --> OR

- <logicalOp>.syn = new OpNode(OR.tkn);

87. <relationalOp> --> LT
    - <logicalOp>.syn = new OpNode(LT.tkn);

88. <relationalOp> --> LE
    - <logicalOp>.syn = new OpNode(LE.tkn);

89. <relationalOp> --> GT
    - <logicalOp>.syn = new OpNode(GT.tkn);

90. <relationalOp> --> GE
    - <logicalOp>.syn = new OpNode(GE.tkn);

91. <relationalOp> --> EQ
    - <logicalOp>.syn = new OpNode(EQ.tkn);

92. <relationalOp> --> NE
    - <logicalOp>.syn = new OpNode(NE.tkn);

93. <declareStmt> --> DECLARE <idList> COLON <dataType> SEMICOL
    - <declareStmt>.syn = new DeclareNode(<dataType>.syn,<idList>.syn);

94. <conditionalStmt> --> SWITCH BO ID BC START <caseStmts><Default> END
    - <conditionalStmt>.syn = new ConditionalNode(ID.tkn, <caseStmts>.syn, <Default>.syn);

95. <caseStmts> --> CASE <value> COLON <statements> BREAK SEMICOL <N9>
    - <statements>.inh = NULL;
    - <N9>.inh = new CaseNode(<value>.syn,<statements>.syn);
    - <caseStmts>.syn = <N9>.syn;

96. <N9> --> CASE <value> COLON <statements> BREAK SEMICOL <N9%>
    - <statements>.inh = NULL;
    - <N9%>.inh = new CaseNode(<value>.syn,<statements>.syn);
    - insertAtBeginning(<N9>.inh, <N9%>.syn);
    - <N9>.syn = <N9%>.syn;

97. <N9> --> #
    - <N9>.syn = <N9>.inh;

98. <value> --> NUM
    - <values>.syn = new NumNode(NUM.tkn);

99. <value> --> TRUE
    - <value>.syn = new BoolNode(TRUE.tkn);

100. <value> --> FALSE

- <value>.syn = new BoolNode(FALSE.tkn);

101. <Default> --> DEFAULT COLON <statements> BREAK SEMICOL

- <statements>.inh = NULL;
- <Default>.syn = <statements>.syn;

102. <Default> --> #

- <Default>.syn = new NullNode();

103. <range> --> NUM RANGEOP NUM%

- <range>.syn = new RangeNode(NUM.tkn, NUM%.tkn);