Semantic Rules for AST Creation

Group 2

Sanjeet Kapadia 2018B4A70137P
Lingesh Kumaar 2018B4A70857P
Aman Mishra 2018B4A70877P
Sidharth Varghese 2019A7PS1133P
Edara Bala Mukesh 2019A7PS0081P

1. <program> => <otherFunctions> <mainFunction>
   - <program>.treeNode = makeNodeChildren(<otherFunctions>.treeNode, <mainFunction>.treeNode);

2. <mainFunction> => TK_MAIN <stmts> TK_END
   - <mainFunction>.treeNode = makeTreeNode('main', <stmts>.treeNode);

3. <otherFunctions> => <function> $<otherFunctions>_1$
   - <otherFunctions>.treeNode = makeTreeNodeList(head = <function>.treeNode, tail = $<otherFunctions>_1$.treeNode);

4. <otherFunctions> => ε
   - <otherFunctions>.treeNode = NULL;

5. <function> => TK_FUNID <input_par> <output_par> TK_SEM <stmts> TK_END
   - <function>.treeNode = makeTreeNode(TK_FUNID.value, <input_par>.treeNode, <output_par>.treeNode, <stmts>.treeNode);

6. <input_par> => TK_INPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR
   - <input_par>.treeNode = <parameter_list>.treeNode;

7. <output_par> => TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR
   - <output_par>.treeNode = <parameter_list>.treeNode;

8. <output_par> => ε
   - <output_par>.treeNode = NULL;

9. <parameter_list> => <dataType> TK_ID <remaining_list>
   - newnode = makeTreeNode(<dataType>.name, TK_ID.value);
   - <parameter_list>.treeNode = makeTreeNodeList(head = newnode, tail = <remainingList>.treeNode);

10. <dataType> => <primitiveDatatype>
    - <dataType>.name = <primitiveDatatype>.name;

11. <dataType> => <constructedDatatype>
    - <dataType>.name = <constructedDatatype>.name;

12. <primitiveDatatype> => TK_INT
    - <primitiveDatatype>.name = ‘int’;

13. <primitiveDatatype> => TK_REAL
    - <primitiveDatatype>.name= ‘real’;

14. <constructedDatatype> => TK_RECORD TK_RUID
    - <constructedDatatype>.name = TK_RUID.name;

15. <constructedDatatype> => TK_UNION TK_RUID
    - <constructedDatatype>.name = TK_RUID.name;

16. <constructedDatatype> => TK_RUID
    - <constructedDatatype>.name = TK_RUID.name;

17. <remaining_list> => TK_COMMA <parameter_list>
    - <remaining_list>.treeNode = <parameter_list>.treeNode;

18. <remaining_list> => ε
    - <remaining_list>.treeNode = NULL;

19. <stmts> => <typeDefinitions> <declarations> <otherStmts> <returnStmt>
    - <stmts>.treeNode = makeTreeNode(<typeDefinitions>.treeNode, <declarations>.treeNode, <otherStmts>.treeNode, <returnStmt>.treeNode);

20. <typeDefinitions>=> <actualOrRedefined> $<typeDefinitions>_1$
    - <typeDefinitions>.treeNode = makeTreeNodeList(head = < actualOrRedefined>.treeNode, tail = $<typeDefinitions>_{1.}$treeNode);

21. <typeDefinitions> => ε
    - <typeDefinitions>.treeNode = NULL;

22. <actualOrRedefined> => <typeDefinition>
    - <actualOrRedefined>.treeNode =<typeDefinition>.treeNode;

23. <actualOrRedefined> => <definetypestmt>
    - <actualOrRedefined>.treeNode =< definetypestmt >.treeNode;

24. <typeDefinition> => TK_RECORD TK_RUID <fieldDefinitions> TK_ENDRECORD
    - <typeDefinition>.treeNode = makeTreeNode(TK_RUID.name, <fieldDefinitions>.treeNode);

25. <typeDefinition> => TK_UNION TK_RUID <fieldDefinitions> TK_ENDUNION

- <typeDefinition>.treeNode = makeTreeNode(TK_RUID.name, <fieldDefinitions>.treeNode);

26. <fieldDefinitions> => <fieldDefinition>$_1$ <fieldDefinition>$_2$ <moreFields>
    - newnode = makeTreeNodeList(head = <fieldDefinition>$_2$ .treeNode, tail = <moreFields>.treeNode);
    - <fieldDefinitions>.treeNode = makeTreeNodeList(head = <fieldDefinition>$_1$.treeNode, tail = newnode);

27. <fieldDefinition> => TK_TYPE <fieldType> TK_COLON TK_FIELDID TK_SEM
    - <fieldDefinition>.treeNode = makeTreeNode(<fieldType>.name , TK_FIELDID.name);

28. <fieldType>=> <primitiveDatatype>
    <fieldType>.name = <primitiveDatatype>.name

29. <fieldType>=> TK_RUID
    <fieldType>.name = TK_RUID.name

30. <moreFields> => <fieldDefinition> <moreFields>$_1$
    - <moreFields>.treeNode = makeTreeNodeList(head = <fieldDefinition>.treeNode, tail = <moreFields>$_1$.treeNode);

31. <moreFields> => ε
    - <moreFields>.treeNode = NULL;

32. <declarations> => <declaration> <declarations>$_1$
    - <declarations>.treeNode = makeTreeNodeList(head = <declaration>.treeNode, tail = <declarations>$_1$.treeNode);

33. <declarations> => ε
    - <declarations>.treeNode = NULL;

34. <declaration> => TK_TYPE <dataType> TK_COLON TK_ID <global_or_not> TK_SEM
    - <declaration>.treeNode = makeTreeNode(<dataType>.name, TK_ID.value, <global_or_not>.flag);

35. <global_or_not> => TK_COLON TK_GLOBAL
    - <global_or_not>.flag = True;

36. <global_or_not> => ε
    - <global_or_not>.flag = False;

37. <otherStmts> => <stmt> <otherStmts>
    - <otherStmts>.treeNode = makeTreeNode(<stmt>.treeNode, <otherStmts>.treeNode);

38. <otherStmts> => ε
    - <otherStmts>.treeNode = NULL;

39. <stmt> => <assignmentStmt>
    - <stmt>.treeNode = <assignmentStmt>.treeNode;

40. <stmt> => <iterativeStmt>
    - <stmt>.treeNode = <iterativeStmt>.treeNode;

41. <stmt> => <conditionalStmt>
    - <stmt>.treeNode = <conditionalStmt>.treeNode;

42. <stmt> => <ioStmt>
    - <stmt>.treeNode = <ioStmt>.treeNode;

43. <stmt> => <funCallStmt>
    - <stmt>.treeNode = <funCallStmt>.treeNode;

44. <assignmentStmt> => <SingleOrRecId> TK_ASSIGNOP <arithmeticExpression> TK_SEM
    - <assignmentStmt>.treeNode = makeTreeNode('<---', <SingleOrRecId>.treeNode, <arithmeticExpression>.treeNode);

45. <SingleOrRecId> => TK_ID <option_single_constructed>
    - <SingleOrRecId>.treeNode = makeTreeNode(TK_ID.value, <option_single_constructed>.treeNode);

46. <option_single_constructed> => <oneExpansion> <moreExpansions>
    - < option_single_constructed>.treeNode = makeTreeNodeList(head = <oneExpansion>.treeNode, tail = < moreExpansions >.treeNode);

47. <option_single_constructed> => ε
    <option_single_constructed>.treeNode = NULL;

48. <oneExpansion> => TK_DOT TK_FIELDID
    <oneExpansion>.treeNode = TK_FIELDID.name

49. <moreExpansions> => <oneExpansion> <moreExpansions>
    <moreExpansions>.treeNode = makeTreeNodeList(head = <oneExpansion>.treeNode, tail = <moreExpansions>.treeNode);

50. <moreExpansions> => ε
    <moreExpansions>.treeNode = NULL;

51. <funCallStmt> => <outputParameters> TK_CALL TK_FUNID TK_WITH TK_PARAMETERS <inputParameters> TK_SEM
    - <funCallStmt>.treeNode = makeTreeNode(TK_FUNID.value, <outputParameters>.treeNode, <inputParameters>.treeNode);

52. <outputParameters> => TK_SQL <idList> TK_SQR TK_ASSIGNOP
    - <outputParameters>.treeNode = makeTreeNode('<---', <idList>.treeNode);

53. <outputParameters> => ε
    - <outputParameters>.treeNode = NULL;

54. <inputParameters> => TK_SQL <idList> TK_SQR
    - <inputParameters>.treeNode = <idList>.treeNode;

55. <iterativeStmt> => TK_WHILE TK_OP <booleanExpression> TK_CL <stmt> <otherStmts> TK_ENDWHILE
    - newnode = makeTreeNodeList(head = <stmt>.treeNode, tail = <otherStmts>.treeNode);
    - <iterativeStmt>.treeNode = makeTreeNode('while', <booleanExpression>.treeNode, newnode);

56. <conditionalStmt> => TK_IF TK_OP <booleanExpression> TK_CL TK_THEN <stmt> <otherStmts> <elsePart>
    - newnode = makeTreeNodeList(head = <stmt>.treeNode, tail = <otherStmts>.treeNode);
    - <conditionalStmt> = makeTreeNode('if', <booleanExpression>.treeNode, newnode, <elsePart>.treeNode);

57. <elsePart> => TK_ELSE <stmt> <otherStmts> TK_ENDIF
    - newnode = makeTreeNodeList(head = <stmt>.treeNode, tail = <otherStmts>.treeNode);
    - <elsePart>.treeNode = makeTreeNode('else', newnode);

58. <elsePart> => TK_ENDIF
    - <elsePart>.treeNode = NULL;

59. <ioStmt> => TK_READ TK_OP <var> TK_CL TK_SEM
    - <ioStmt>.treeNode = makeTreeNode('read', <var>.treeNode);

60. <ioStmt> => TK_WRITE TK_OP <var> TK_CL TK_SEM
    - <ioStmt>.treeNode = makeTreeNode('write', <var>.treeNode);

61. <arithmeticExpression> => <term> <expPrime>
    - <arithmeticExpression>.treeNode = <expPrime>.syn;
    - <expPrime>.inh = <term>.treeNode;

62. <expPrime> => <lowPrecedenceOperators> <term> $<expPrime>_1$
    - $<expPrime>_1$.inh = makeTreeNode(<lowPrecedenceOperators>.name, <expPrime>.inh, <term>.treeNode);
    - <expPrime>.syn = $<expPrime>_1$.syn;

63. <expPrime> => ε
    - <expPrime>.syn = <expPrime>.inh;

64. <term> => <factor> <termPrime>
    - <term>.treeNode = <termPrime>.syn;
    - <termPrime>.inh = <factor>.treeNode;

65. <termPrime> => <highPrecedenceOperators> <factor> <termPrime>$_1$
    - <termPrime>$_1$.treeNode = makeTreeNode(<highPrecedenceOperators>.name, <termPrime>.inh, <factor>.treeNode);
    - <termPrime>.syn = <termPrime>$_1$.syn;

66. <termPrime> => ε
    - <termPrime>.syn = <termPrime>.inh;

67. <factor> => TK_OP <arithmeticExpression> TK_CL
    - <factor>.treeNode = <arithmeticExpression>.treeNode;

68. <factor> => <var>
    - <factor>.treeNode = <var>.treeNode;

69. <highPrecedenceOperators> => TK_MUL
    - <highPrecedenceOperators>.name = '*';

70. <highPrecedenceOperators> => TK_DIV
    - <highPrecedenceOperators>.name = '/';

71. <lowPrecedenceOperators> => TK_PLUS
    - <lowPrecedenceOperators>.name = '+';

72. <lowPrecedenceOperators> => TK_MINUS
    - <lowPrecedenceOperators>.name = '-';

73. <booleanExpression> => TK_OP <booleanExpression>$_1$ TK_CL <logicalOp> TK_OP <booleanExpression>$_2$ TK_CL
    - <booleanExpression>.treeNode = makeTreeNode(<logicalOp>.name, <booleanExpression>$_1$.treeNode, <booleanExpression>$_2$.treeNode);

74. <booleanExpression> => <var>$_1$ <relationalOp> <var>$_2$
    - <booleanExpression>.treeNode = makeTreeNode(<relationalOp>.treeNode, <var>$_1$.treeNode, <var>$_2$.treeNode);

75. <booleanExpression> => TK_NOT TK_OP <booleanExpression>$_1$ TK_CL
    - <booleanExpression>.treeNode = makeTreeNode('~', <booleanExpression>$_1$.treeNode);

76. <var> => <singleOrRecId>
    - <var>.treeNode =<singleOrRecId>.treeNode;

77. <var> => TK_NUM
    - <var>.treeNode = makeTreeNode(TK_NUM.value);

78. <var> => TK_RNUM
    - <var>.treeNode = makeTreeNode(TK_RNUM.value);

79. <logicalOp> => TK_AND
    - <logicalOp>.name = '&&&';

80. <logicalOp> => TK_OR
    - <logicalOp>.name = '@@@';

81. <relationalOp> => TK_LT
    - <relationalOp>.name = '<';

82. <relationalOp> => TK_LE
    - <relationalOp>.name = '<=';

83. <relationalOp> => TK_EQ
    - <relationalOp>.name = '==';

84. <relationalOp> => TK_GT
    - <relationalOp>.name = '>';

85. <relationalOp> => TK_GE
    - <relationalOp>.name = '>=';

86. <relationalOp> => TK_NE
    - <relationalOp>.name = '!=';

87. <returnStmt> => TK_RETURN <optionalReturn> TK_SEM
    - <returnStmt>.treeNode = <optionalReturn>.treeNode;

88. <optionalReturn> => TK_SQL <idList> TK_SQR
    - <optionalReturn>.treeNode = <idList>.treeNode;

89. <optionalReturn> => ε
    - <optionalReturn>.treeNode = NULL;

90. <idList> => TK_ID <more_ids>
    - <idList>.treeNode = makeTreeNodeList(head = TK_ID.value, tail = <more_ids>.treeNode);

91. <more_ids> => TK_COMMA <idList>

- <more_ids>.treeNode = <idList>.treeNode;

92. <more_ids> => ε
    - <more_ids>.treeNode = NULL;

93. <definetypestmt> => TK_DEFINETYPE <recOrUnion> TK_RUID_1 TK_AS TK_RUID_2
    <definetypestmt>.treeNode = makeTreeNode('definetype', <recOrUnion>.name, TK_RUID_1.name, TK_RUID_2.name);

94. <recOrUnion> => TK_RECORD
    <recOrUnion>.name = 'record'

95. <recOrUnion> => TK_UNION
    <recOrUnion>.name = 'union'