Semantic Rules for AST Creation

Group Number 15

Aadit Deshpande	2019A7PS0077P
Nandan B. Parikh	2019A7PS0097P
Preetika Verma	2019A7PS0088P
Pritika Ramu	2019A7PS1140P
Sneha	2019A7PS0042P

Attributes	***Purpose***
ptr	Pointer to AST node (*Synthesized)*
type	The type of the instance (variable) (*Synthesized)*
name	The string for tokens like TK_RUID and numeric value for tokens like TK_NUM (*Synthesized)*
is_global	Takes True/False values (*Synthesized)*
inh	*Inherited* Attributes for the Arithmetic Expression

S.No.	Production Rule	Semantic Rule
1	<program> -> <otherFunctions> <mainFunction>	<program>.ptr = mkNode("Program", <otherFunctions>.ptr, <mainFunction>.ptr)
2	<mainFunction> -> TK_MAIN <stmts> TK_END	<mainFunction>.ptr = mkNode("Main", <stmts>.ptr)
3	<otherFunctions> -> <function> <otherFunctions> <otherFunctions> -> EPSILON	<otherFunctions>.ptr = mkNode("Function_Sequence", <function>.ptr, <otherFunctions>**_1**.ptr) <otherFunctions>.ptr = NULL
4	<function> ->TK_FUNID <input_par> <output_par> TK_SEM <stmts> TK_END	<function>.ptr = mkNode(TK_FUNID.name, <input_par>.ptr, <output_par>.ptr, <stmts>.ptr**)**
5	<input_par> ->TK_INPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR	<input_par>.ptr = <parameter_list>.ptr
6	<output_par> ->TK_OUTPUT TK_PARAMETER TK_LIST TK_SQL <parameter_list> TK_SQR	<output_par> = <parameter_list>.ptr

	<output_par> -> EPSILON	<output_par>.ptr = NULL
7	<parameter_list> -> <dataType> TK_ID <remaining_list>	<parameter_list>.ptr = mkNode("Parameter_List",<dataType>.type,TK_ID.name, <remaining_list>.ptr)
8	<dataType> -> <primitiveDatatype> <dataType> -> <constructedDatatype>	<dataType>.type = <primitiveDatatype>.type <dataType>.type = <constructedDatatype>.type
9	<primitiveDatatype> -> TK_INT <primitiveDatatype> -> TK_REAL	<primitiveDatatype>.type = "Integer" <primitiveDatatype>.type = "Real"
10	<constructedDatatype> -> TK_RECORD TK_RUID | TK_UNION TK_RUID | TK_RUID	<constructedDatatype>.type = TK_RUID.type
11	<remaining_list> ->TK_COMMA <parameter_list> <remaining_list> -> EPSILON	<remaining_list>.ptr = <parameter_list>.ptr <remaining_list>.ptr = NULL
12	<stmts> -> <typeDefinitions> <declarations> <otherStmts> <returnStmt>	<stmts>.ptr = mkNode("Statements", <typeDefinitions>.ptr, <declarations>.ptr, <otherStmts>.ptr, <returnStmt>.ptr)
13	<typeDefinitions> -> <actualOrRedefined> <typeDefinitions> <typeDefinitions> ->EPSILON	<typeDefinitions>.ptr = mkNode("Type_Definition_Sequence", <actualOrRedefined>.ptr, <typeDefinitions>**_1**.ptr) <typeDefinitions>.ptr = NULL
14	<actualOrRedefined> -> <typeDefinition> <actualOrRedefined> -> <definetypestmt>	<actualOrRedefined>.ptr = <typeDefinition>.ptr <actualOrRedefined>.ptr = <definetypestmt>.ptr
15	<typeDefinition> -> TK_RECORD TK_RUID <fieldDefinitions> TK_ENDRECORD	<typeDefinition>.ptr = mkNode("Record_Type_Definition", TK_RUID.name, <fieldDefinitions>.ptr)
16	<typeDefinition> -> TK_UNION TK_RUID <fieldDefinitions> TK_ENDUNION	<typeDefinition>.ptr = mkNode("Union_Type_Definition", TK_RUID.name, <fieldDefinitions>.ptr)
17	<fieldDefinitions> -> <fieldDefinition> <fieldDefinition> <moreFields>	<fieldDefinitions>.ptr = mkNode("Field_Definition_Sequence", <fieldDefinition>**_1**.ptr, <fieldDefinition>**_2**.ptr <moreFields>.ptr)
18	<fieldDefinition> -> TK_TYPE <fieldType > TK_COLON TK_FIELDID TK_SEM	<fieldDefinition> -> mkNode("Field_Definition", <fieldType>.type, TK_FIELDID.name)
19	<fieldtype> -> <primitiveDatatype>	<fieldType>.type = <primitiveDatatype>.type

	<fieldtype> -> TK_RUID	<fieldType>.type = TK_RUID.type
20	<moreFields> -> <fieldDefinition> <moreFields> <moreFields> -> EPSILON	<moreFields>.ptr = mkNode(“More_Fields”, <fieldDefinition>.ptr, <moreFields>.ptr) <moreFields>.ptr = NULL
21	<declarations> -> <declaration> <declarations> <declarations> -> EPSILON	<declarations>.ptr = mkNode(“Declaration_Sequence”, <declaration>.ptr. <declarations>.ptr) <declarations>.ptr = NULL
22	<declaration> -> TK_TYPE<dataType> TK_COLON TK_ID <global_or_not> TK_SEM	<declaration>.ptr =mkNode(“Declaration”, <dataType>.type, TK_ID.name, <global_or_not>.is_global)
23	<global_or_not> -> TK_COLON TK_GLOBAL <global_or_not> -> EPSILON	<global_or_not>.is_global = True <global_or_not>.is_global = False
24	<otherStmts> -> <stmt> <otherStmts> <otherStmts> ->EPSILON	<otherStmts>.ptr = mkNode(“Other_Statements”, <stmt>.ptr, <otherStmts>.ptr) <otherStmts>.ptr = NULL
25	<stmt> -> <assignmentStmt> <stmt> -> <iterativeStmt> <stmt> -> <conditionalStmt> <stmt> -> <ioStmt> <stmt> -> <funCallStmt>	<stmt>.ptr = <assignmentStmt>.ptr <stmt>.ptr = <iterativeStmt>.ptr <stmt>.ptr = <conditionalStmt>.ptr <stmt>.ptr = <ioStmt>.ptr <stmt>.ptr = <funCallStmt>.ptr
26	<assignmentStmt> -> <SingleOrRecId> TK_ASSIGNOP <arithmeticExpression> TK_SEM	<assignmentStmt>.ptr = mkNode(“Assignment”, <SingleOrRecId>.ptr, <arithmeticExpression>.ptr)
27	<SingleOrRecId> -> TK_ID <option_single_constructed>	<SingleOrRecId>.ptr = mkNode(“Single_or_Record_ID”, TK_ID.name, <option_single_constructed>.ptr)
28	<option_single_constructed> -> EPSILON <option_single_constructed> -> <oneExpansion> <moreExpansions>	<option_single_constructed>.ptr = NULL <option_single_constructed>.ptr = mkNode(“Option_Single_Constructed”, <oneExpansion>.ptr, <moreExpansions>.ptr)

29	<oneExpansion> -> TK_DOT TK_FIELDID	<oneExpansion>.ptr =mkNode(“One_Expansion”, TK_FIELDID.name)
30	<moreExpansions> -> <oneExpansion> <moreExpansions> <moreExpansions> -> EPSILON	<moreExpansions>.ptr = mkNode(“More_Expansions” <oneExpansion>.ptr, <moreExpansions>.ptr) <moreExpansions>.ptr = NULL
31	<funCallStmt> -> <outputParameters> TK_CALL TK_FUNID TK_WITH TK_PARAMETERS <inputParameters> TK_SEM	<funCallStmt>.ptr = mkNode(“Function_Call”, <outputParameters>.ptr, TK_FUNID.name <inputParameters>.ptr)
32	<outputParameters> -> TK_SQL <idList> TK_SQR TK_ASSIGNOP <outputParameters>-> EPSILON	<outputParameters>.ptr = <idList>.ptr <outputParameters>.ptr = NULL
33	<inputParameters>-> TK_SQL <idList> TK_SQR	<inputParameters>.ptr = <idList>.ptr
34	<iterativeStmt>-> TK_WHILE TK_OP <booleanExpression> TK_CL <stmt><otherStmts> TK_ENDWHILE	<iterativeStmt>.ptr = mkNode(“Iterative”, <booleanExpression>.ptr, <stmt>.ptr, <otherStmts>.ptr)
35	<conditionalStmt>-> TK_IF TK_OP <booleanExpression> TK_CL TK_THEN <stmt><otherStmts> <elsePart>	<conditionalStmt>.ptr = mkNode(“Conditional”, <booleanExpression>.ptr, <stmt>.ptr, <otherStmts>.ptr, <elsePart>.ptr)
36	<elsePart>->TK_ELSE <stmt><otherStmts> TK_ENDIF <elsePart>->TK_ENDIF	<elsePart>.ptr = mkNode(“Else_Part”, <stmt>.ptr, <otherStmts>.ptr) <elsePart>.ptr = NULL
37	<ioStmt>->TK_READ TK_OP <var> TK_CL TK_SEM <ioStmt>->TK_WRITE TK_OP <var> TK_CL TK_SEM	<ioStmt>.ptr = mkNode(“I/O_Read”, <var>.ptr) <ioStmt>.ptr = mkNode(“I/O_Write”, <var>.ptr)
38	<arithmeticExpression> -> <term> <expPrime>	<arithmeticExpression>.ptr = <expPrime>.ptr <expPrime>.inh = <term>.ptr
39	<expPrime> -> <lowPrecedenceOperators> <term> <expPrime>**_1** <expPrime>-> EPSILON	<expPrime>**_1**.inh = mkNode(<lowPrecedenceOperators>.name, <expPrime>.inh, <term>.ptr) <expPrime>.ptr = <expPrime>**_1**.ptr

40	<term>-> <factor> <termPrime>	<term>.ptr = <termPrime>.ptr <termPrime>.inh = <factor>.ptr
41	<termPrime> -> <highPrecedenceOperators><factor> <termPrime> <termPrime>->EPSILON	<termPrime>**_1**.inh = mkNode(<highPrecedenceOperators>.name, <termPrime>.inh, <factor>.ptr) <termPrime>.ptr = <termPrime>**_1**.ptr
42	<factor> -> TK_OP <arithmeticExpression> TK_CL <factor>-><var>	<factor>.ptr = <arithmeticExpression>.ptr <factor>.ptr = <var>.ptr
43	<highPrecedenceOperator>-> TK_MUL <highPrecedenceOperator>->TK_DIV	<highPrecedenceOperator>.name = “MUL” <highPrecedenceOperator>.name = “DIV”
44	<lowPrecedenceOperators> -> TK_PLUS <lowPrecedenceOperators>->TK_MINUS	<lowPrecedenceOperators>.name = “PLUS” <lowPrecedenceOperators>.name = “MINUS”
45	<booleanExpression>->TK_OP <booleanExpression> TK_CL <logicalOp> TK_OP <booleanExpression> TK_CL	<booleanExpression>.ptr = mkNode(<logicalOp>.name**,** <booleanExpression>**_1**.ptr, <booleanExpression>**_2**.ptr)
46	<booleanExpression>-> <var> <relationalOp> <var>	<booleanExpression>.ptr = mkNode(<relationalOp>.name, <var>_1.ptr, <var>_2.ptr)
47	<booleanExpression>-> TK_NOT TK_OP <booleanExpression> TK_CL	<booleanExpression>-> mkNode(“NOT”, <booleanExpression>_1.ptr)
48	<var>-> <singleOrRecId> <var>->TK_NUM <var>-> TK_RNUM	<var>.ptr = <singleOrRecId>.ptr <var>.ptr = mkLeaf(“Variable_Number”, TK_NUM.name) <var>.ptr = mkLeaf(“Variable_Real_Number”, TK_RNUM.name)
49	<logicalOp>->TK_AND <logicalOp>->TK_OR	<logicalOp>.name = “AND” <logicalOp>.name = **“**OR”
50	<relationalOp>-> TK_LT <relationalOp>-> TK_LE <relationalOp>-> TK_EQ	<relationalOp>.name = “LT” <relationalOp>.name = “LE” <relationalOp>.name = “EQ”

	<relationalOp>->TK_GT <relationalOp>->TK_GE <relationalOp>->TK_NE	<relationalOp>.name = “GT” <relationalOp>.name = “GE” <relationalOp>.name = “NE”
51	<returnStmt>->TK_RETURN <optionalReturn> TK_SEM	<returnStmt>.ptr = mkNode(“Return_Statement”, <optionalReturn>.ptr)
52	<optionalReturn>->TK_SQL <idList> TK_SQR <optionalReturn>-> EPSILON	<optionalReturn>.ptr = mkNode(“Return_Parameters”, <idList>.ptr) <optionalReturn>.ptr = NULL
53	<idList>-> TK_ID <more_ids>	<idList>.ptr = mkNode(“ID_List”, TK_ID.name, <more_ids>.ptr)
54	<more_ids>-> TK_COMMA <idList> <more_ids>-> EPSILON	<more_ids>.ptr = <idList>.ptr <more_ids>.ptr = NULL
55	<definetypestmt>->TK_DEFINETYPE <A> TK_RUID TK_AS TK_RUID TK_SEM	<definetypestmt>.ptr = mkNode(“Define_Type”, <A>.type, TK_RUID_1.name, TK_RUID_2.name)
56	<A> -> TK_RECORD <A> -> TK_UNION	<A>.type = “RECORD” <A>.type = “UNION”