## Attribute Grammar

### Attributes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Symbol | Attribute Name | Java Type | Inherited/Synthesized | Description |
| Expression | type | Type | Synthetic | The type of the expression |
| Expression | lvalue | Boolean | Synthetic | Wether the expression can be the assigned in the assignment or not |
|  |  |  |  |  |

### Auxiliary Functions

|  |  |
| --- | --- |
| Name | Description |
| sameType(type+) | Returns true if types are equal |
| isInteger(type+) | Returns true if types are integer |
| isReal(type+) | Returns true if types are real |
| isStructType(type+) | Returns true if types are struct |

### Rules

|  |  |  |
| --- | --- | --- |
| Node | Predicates | Semantic Functions |
| program → definition\* |  |  |
| varDefinition:definition → name:string type |  |  |
| structDefinition:definition → name:string field\* |  |  |
| functionDefinition:definition → name:string varDefinition\* type? definition\* statement\* |  |  |
| field → name:string type |  |  |
| print:statement → expression |  |  |
| read:statement → expression |  |  |
| functionCallStatement:statement → name:string expression\* |  |  |
| assignment:statement → left:expression right:expression | sameType(left.type, right.type)  left.lvalue == true |  |
| conditional:statement → expression ifStatements:statement\* elseStatements:statement\* |  |  |
| while:statement → expression loopStatements:statement\* |  |  |
| return:statement → expression? |  |  |
| intType:type → ε |  |  |
| floatType:type → ε |  |  |
| charType:type → ε |  |  |
| arrayType:type → intValue:int type |  |  |
| structType:type → name:string |  |  |
| variable:expression → name:string |  | variable.lvalue = true |
| intLiteral:expression → intValue:int |  |  |
| floatLiteral:expression → floatValue:float |  |  |
| charLiteral:expression → charValue:char |  |  |
| functionCallExpression:expression → name:string expression\* |  |  |
| structAccess:expression → expression name:string |  | array.lvalue=true |
| arrayAccess:expression → left:expression right:expression | isInteger(right.type) | array.lvalue=true |
| cast:expression → type expression |  | cast.type = type  expression.type = type  arithmetic.lvalue=false |
| arithmetic:expression → left:expression operator:string right:expression | operator.equals(‘+-\*/’) && isInteger(left.type, right.type)  operator.equals(‘%’)  && isReal(left.type, right.type) | arithmetic.lvalue = false  arithmetic.type = left.type |
| Logic:expression → left:expression operator:string right:expression |  |  |
| logicAnd:expression → left:expression right:expression |  |  |
| logicOr:expression → left:expression right:expression |  |  |
| logicNot:expression → expression |  |  |

Operators samples (cut & paste if needed):  
⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ∩ ⊂ ⊄ ∑ ∃ ∀