BoogiePl Grammar

Program	::=	*Decl**
Decl		*Type*|*Constant*|*Function*|*Axiom*|*Variable* |*Procedure*|*Implementation*
Type	::=	**bool**|**int**|**ref**|**name** |*Id*|**any** |"["*"Type"*"]" *Type* |"["*"Type"*", *"Type"*"]" *Type*
Type	::=	**type** *IdList* ";"
IdList	::=	*Id*["," *IdList*]
Constant	::=	**Const** *IdTypeList* ";"
IdTypeList	::=	*IdType*[";" *IdType*]
IdType	::=	*Id* ":" *Type*
Function	::=	**function** *IdList* "(" [*OptIdTypeList*] ")" **returns** "(" *OptIdType* ")" ";"
OptIdType	::=	[*Id* ":"] *Type*
OptIdTypeList	::=	*OptIdType* ["," *OptIdTypeList*]
Axiom	::=	**axiom** *Expression* ";"
Variable	::=	**var** *IdTypeList* ";"
Procedure	::=	**procedure** *Id Signature* ";" *Spec** | **procedure** *Id Signature Spec* Body*
Signature	::=	*ParamList* [**returns** *ParamList*]
ParamList	::=	"(" [*IdTypeList*] ")"
Spec	::=	**requires** *Expression* ";" |**modifies** [*IdList*] ";" |**ensures** *Expression* ";"
Implementation	::=	**implementation** *Id Signature Body*
Body	::=	"{" *LocalVarDecl* Block+* "}"
LocalVarDecl	::=	**var** *IdTypeList* ";"
Block	::=	*Id* ":" *Command* TocManifesto*
TocManifesto	::=	**goto** *IdList*";" | **return** ";"
Command	::=	*Id*":="*Expression* ";" | *Id Index* ":="*Expression* ";" | **assert** *Expression* ";" | **assume** *Expression* ";" | **havoc** *IdList* ";" | **call** [*IdList* ":="] *Id* "(" [*ExpressionList*] ")" ";"
Index	::=	"[" *Expression* "]" | "[" *Expression* "," *Expression* "]"
ExpressionList	::=	*Expression* ["," *ExpressionList*]
Expression	::=	*Equivalence*
Equivalence	::=	*Implication* ["<==>"*Equivalence*]
Implication	::=	*LogicalExpr* ["==>"*Implication*]
LogicalExpr	::=	*Relation* | *Relation* "&&" *AndExpr* | *Relation* "||" *OrExpr*
AndExpr	::=	*Relation* ["&&" *AndExpr*]
OrExpr	::=	*Relation* ["||" *OrExpr*]
Relation	::=	*Term* [*RelOp Term*]
Term	::=	[*Term AddOp*] *Factor*
Factor	::=	[*Factor MulOp*] *UnaryExpr*
UnaryExpr	::=	*ArrayExpr* | "!" *UnaryExpr* | "-" *UnaryExpr*
ArrayExpr	::=	*Atom* | *ArrayExpr Index*
Atom	::=	**false** | **true**| **null** | **0** | **1** | **2** | … | *Id* | *Id*"(" [*ExpressionList*] ")" | **old** "(" *Expression* ")" | **cast** "(" *Expression* "," *Type* ")" | *Quantification* | "(" *Expression* ")"
Quantification	::=	"(" *QuantOp IdTypeList* "::" *Expression* ")"
RelOp	::=	"==" | "!=" | "<" | "<=" | ">=" | ">" | "<:"
AddOp	::=	"+" | "-"
MulOp	::=	"*"| "/"| "%"
QuantOp	::=	"forall"| "exists"