8. Expressions

An *expression* denotes a rule of computation that yields a value when the expression is evaluated, except when the expression activates a function and that activation is terminated by a goto statement (see Sections 9.1.3 and 10). The value that is yielded depends upon the values of the constants, bounds, and variables in the expression and also upon the operators and functions that the expression invokes.

Expression= *SimpleExpression* [*RelationalOperator SimpleExpression*].

SimpleExpression = [*Sign*] *Term* { *AddingOperator Term* } .

Term = *Factor* { *MultiplyingOperator Factor* } .

Factor = *UnsignedConstant* | *BoundIdentifier* | *Variable* |
SetConstructor | *FunctionDesignator* |
"not" *Factor* | "(" *Expression* ")" .

UnsignedConstant = *UnsignedNumber* | *CharacterString* |
ConstantIdentifier | "nil" .

SetConstructor = "[" [*ElementDescription* { ","
ElementDescription }] "]" .

ElementDescription = *OrdinalExpression* [".." *OrdinalExpression*] .

FunctionDesignator = *FunctionIdentifier* [*ActualParameterList*] .

RelationalOperator = "=" | "<>" | "<" | "<=" | ">" | ">=" / "in" .

AddingOperator = "+" | "-" | "or" .

MultiplyingOperator = "*" | "/" / "div" | "mod" / "and" .

An ordinal expression is an expression that possesses an ordinal type. A Boolean expression or integer expression is an ordinal expression that possesses the type Boolean or Integer, respectively.

OrdinalExpression = *Expression* .

BooleanExpression = *OrdinalExpression* .

IntegerExpression = *OrdinalExpression* .

8.1. Operands

A multiplying operator in a term has two operands: the part of the term that precedes the operator, and the factor that immediately follows the operator. An adding operator in a simple expression has two operands: