Oberon-0 Semantics

The following summarizes some of the semantic rules of Oberon-0 that need to be checked during or after parsing.

1. Name Declaration

All names (identifiers) in the same scope must be unique, regardless of whether they refer to constants, types, variables, or procedures. Duplicate declarations must be reported.

1.1. Constants

The expression must evaluate to a constant.

1.2. Arrays

The expression must be constant and evaluate to a number larger than `0`. The specified type must exist.

1.3. Records

A record defines a new scope. The names of the declared record fields must be unique in this scope. The specified types must exist.

1.4. Variables

The specified types must exist.

1.5. Procedures

A procedure defines a new scope. All names declared in the procedure must be unique in this scope. Formal parameters that have a structured type (`ARRAY` or `RECORD`) cannot be `VAR`-parameters. Values of structured types are always passed by reference.

2. Expressions

Expressions must by type-checked and use of invalid types must be reported. Constant expressions must be evaluated.

2.1. Arithmetic Operators

Both arguments must be `INTEGER` values, the result is of type `INTEGER`.

2.2. Comparison Operators

Both arguments must have the same type, the result is of type `BOOLEAN`.

2.3. Boolean Operators

Arguments must be `BOOLEAN` values, the result is of type `BOOLEAN`.

3. Name Use

Names must be declared before they can be used. Undeclared names must be reported.

3.1. Variables and Actual Parameters

If a selector is present, it must be checked that the name refers to either an `ARRAY` or `RECORD`, respectively. If a selector references a `RECORD`-field, existence of this field must be checked. If a selector indexes an `ARRAY`-field and the index is given as a constant or constant expression, array bounds must be checked.

3.2. Procedure Calls

The number and types of the actual parameters must match the number and types of the formal parameters. Constant values and expressions cannot be passed as `VAR`-parameters. As a default, parameters are passed by value.

4. Statements

4.1. Assignments

The name on the left-hand side of the assignment must refer to a variable. The type of expression that is assigned must be compatible with the type of the variable.

4.2. IF-Statements

The type of expression representing the condition must of type `BOOLEAN`.

4.3. WHILE-Statements

The type of expression representing the condition must of type `BOOLEAN`.