## 6.2 Structured types and values

The **type** keyword is also used to specify structured types such as **record** types, **record** **of** types, **set** types, **set** **of** types, **enumerated** types and **union** types.

Values of these types may be given using an explicit assignment notation or a short-hand value list notation.

EXAMPLE 1:

**const** MyRecordType MyRecordValue:= //assignment notation

{

field1 := '11001'B,

field2 := **true**,

field3 := "A string"

}

// Or

**const** MyRecordType MyRecordValue:= {'11001'B, **true**, "A string"} //value list notation

The assignment notation can be used for **record, record of**, **set**, **set of** and **union** value notations and for arrays. In this notation each field shall not appear more than once. The value list notation can be used for **record, record of**, **set** and **set of** value notations and for arrays. The index notation can be used for **record of** and **set of** value notations and for arrays. In this notation the number of values is limited by the maximum length that is allowed by the type definition. See more details in the subsequent clauses.

EXAMPLE 2:

**var** MyRecordType MyVariable:= //assignment notation

{

field1 := '11001'B,

// field2 implicitly unspecified

field3 := "A string"

}

// Or  
 **var** MyRecordType MyVariable:= //assignment notation

{

field1 := '11001'B,  
 field2 := -, // field2 explicitly unspecified

field3 := "A string"

}

// Or

**var** MyRecordType MyVariable:= {'11001'B, -, "A string"} //value list notation

It is not allowed to mix the two value notations in the same (immediate) context.

EXAMPLE 3:

// This is disallowed

**const** MyRecordType MyRecordValue:= {MyIntegerValue, field2 := **true**, "A string"}

Where applicable TTCN‑3 type definitions may be recursive. The user, however, shall ensure that all type recursion is resolvable and that no infinite recursion occurs.

In case of record and set types, to avoid infinite recursion, fields referencing to its own type, shall be optional.

EXAMPLE 4:

// Valid recursive record type definition

**type** **record** MyRecord1  
    {   
        FieldType1 field1,  
        MyRecord1 field2 **optional**,

        FieldType3 field3   
    }

// Invalid recursive record type definition causing an error

**type** **record** MyRecord2  
    {   
        FieldType1 field1,  
        MyRecord2 field2,

        FieldType3 field3   
    }

In case of union types, to avoid infinite recursion, at least one of the alternatives shall not reference its own type.

EXAMPLE 5:

// Valid recursive union type definition

**type** **union** MyUnion1

{

MyUnion1 choice1,

**charstring** choice2

}

// Invalid recursive union type definition causing an error

**type** **union** MyUnion2

{

MyUnion2 choice1,

MyUnion2 choice2

}