### 7.5.2 Derivation by list

XSD *list* components shall be mapped to the TTCN-3 **record of** type. In their simplest form, when the i*temType* attribute identifies the base type of the derivation, the replicated type of the TTCN-3 record of shall be the type mapped from the XSD type referenced by *itemType*.

When the XSD *list* is used to derive a list type from an (embedded) unnamed XSD type, first the type included by the *list* start and end tags shall implicitly be translated to TTCN-3 and this type shall be the replicated type of the generated TTCN-3 **record of**.

Finally, the encoding instruction "list" shall be applied to the generated **record of** type.

EXAMPLE 1: Mapping a list derived by using the i*temType* attribute

<xsd:simpleType name=**"**e19**"**>

 <list itemType=**"float"**/>

</xsd:simpleType>

// Will translate to

**type** **record of** XSD.Float E19
**with** **{**
 **variant** "list";

 **variant** "name as uncapitalized"

**}**

EXAMPLE 2: Mapping an unnamed simple union type derived by list:

 <?xml version="1.0" encoding="UTF-8"?>

 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"

 xmlns:tns="http://www.example.org/list\_union"

 targetNamespace="http://www.example.org/list\_union" >

 <xsd:element name="MyUnionList">

 <xsd:simpleType>

 <xsd:list>

 <xsd:simpleType>

 <xsd:union>

 <xsd:simpleType>

 <xsd:restriction base="xsd:boolean" />

 </xsd:simpleType>

 <xsd:simpleType>

 <xsd:restriction base="xsd:float" />

 </xsd:simpleType>

 </xsd:union>

 </xsd:simpleType>

 </xsd:list>

 </xsd:simpleType>

 </xsd:element>

 </xsd:schema>

 Is translated to TTCN-3 as (translation of the embedded union type see in the next clause):

 **module** http\_www\_example\_org\_list\_union {

 **import** **from** XSD all;

 **type** **record of union**

 {

 XSD.Boolean alt\_,

 XSD.Float alt\_1

 } MyUnionList

  **with** {

 **variant** "list";

 **variant** "element";

 **variant** ([-]) "useUnion";

 **variant** ([-].alt\_) "name as ''";

 **variant** ([-].alt\_1) "name as ''";

 };

 }

 **with** {

 **encode** "XML";

 **variant** "namespace as 'http://www.example.org/list\_union' prefix 'tns'";

 **variant** "controlNamespace 'http://www.w3.org/2001/XMLSchema-instance' prefix 'xsi'";

 }

When using any of the supported XSD facets (length, maxLength, minLength) the translation shall follow the mapping for built-in list types, with the difference that the base type shall be determined by an anonymous inner list item type.

EXAMPLE 3: Consider this example:

<xsd:element name="e20">

 <xsd:simpleType>

 <xsd:restriction>

 <xsd:simpleType>

 <xsd:list itemType="float"/>

 </xsd:simpleType>

 <xsd:length value="3"/>

 </xsd:restriction>

 </xsd:simpleType>

</xsd:element>

// Will map to:

**type** **record** **length**(**3**) **of** XSD.Float E20
**with** **{**
 **variant** "name as uncapitalized";

 **variant** "element";
 **variant** "list";

**}**

//For instance the template:

**template** E20 t\_E20:=**{** 1.0, 2.0, 3.0 **}**

// can be encoded, for example, as:

<?xml version="1.0" encoding="UTF-8"?><e20>1.0 2.0 3.0</e20>

The other XSD facets shall be mapped accordingly (refer to respective 6.1 clauses). If no *itemType* is given, the mapping has to be implemented using the given inner type (see clause 7.5.3).

### 7.5.3 Derivation by union

An XSD union is considered as a set of mutually exclusive alternative types for a *simpleType*. As this is compatible with the *union* type of TTCN-3, a *simpleType* derived by *union* in XSD shall be mapped to a union type definition in TTCN-3. The generated TTCN-3 **union** type shall contain one alternative for each member type of the XSD *union*, preserving the textual order of the member types in the initial XSD union type. The field names of the TTCN-3 **union** type shall be the result of applying clause 5.2.2 to either to the unqualified name of the member type (in case of built‑in XSD data types and user defined named types) or to the string "alt" (in case of unnamed member types).

XSD requires (see XML Schema Part 2: Datatypes [9], clause 2.5.1.3) that an element or attribute value of an instance is validated against the member types in the order in which they appear in the XSD definition until a match is found (considering any xsi:type attribute present, see also clause B.3.24). A TTCN-3 tool has to use this strategy as well, when decoding an XSD *union* value.

The encoding instruction "useUnion" shall be applied to the generated **union** type and, in addition, the "name as ''" ("name as followed by a pair of single quote followed by a double quote) encoding instruction shall be applied to each field generated for an unnamed member type.

NOTE: Please note, that alt and the names of several built-in XSD data types are TTCN-3 keywords, hence according to the naming rules these field identifiers will be postfixed with a single underscore character.

EXAMPLE 1: Mapping of named simple type definitions derived by union:

<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.example.org/union"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

*targetNamespace="http://www.example.org/union">*

<xsd:simpleType name="e21memberlist">

 <xsd:union memberTypes="xsd:integer xsd:boolean xsd:string "/>

</xsd:simpleType>

<xsd:element name=*"e21namedElement"* type=*"e21memberlist"*/>

</xsd:schema>

// Results in the following mapping:

**module** http\_www\_example\_org\_union {

**import** **from** XSD **all**;

**type** E21memberlist E21namedElement

**with** {

**variant** "name as uncapitalized";

**variant** "element";

}

**type union** E21memberlist {

 XSD.Integer integer\_,

 XSD.Boolean boolean\_,

 XSD.String string

}

**with {**

 **variant** "name as uncapitalized";

 **variant** "useUnion";

 **variant** (integer\_) "name as 'integer'";

 **variant** (boolean\_) "name as 'boolean'";

}

}

**with** {

**encode** "XML";

**variant** "namespace as 'www.example.org/union'";

**variant** "controlNamespace 'http://www.w3.org/2001/XMLSchema-instance' prefix 'xsi'";

}

// For instance, the below structure:

**template** E21namedElement t\_UnionNamedInt := { integer\_ := 1 }

// will result in the following encoding:

<?xml version="1.0" encoding="UTF-8"?>
<e21namedElement xmlns='www.example.org/union' xmlns:xsd='http://www.w3.org/2001/XMLSchema' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:type= 'xsd:integer'>1</e21namedElement>

EXAMPLE 2: Mapping of unnamed simple type definitions derived by union:

<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.example.org/union">

<!-- Please compare with the previous example -->

<xsd:element name="e21unnamed">

 <xsd:simpleType>

 <xsd:union>

 <xsd:simpleType>

 <xsd:restriction base="xsd:float"/>

 </xsd:simpleType>

 <xsd:simpleType>

 <xsd:restriction base="xsd:integer"/>

 </xsd:simpleType>

 <xsd:simpleType>

 <xsd:restriction base="xsd:string"/>

 </xsd:simpleType>

 </xsd:union>

 </xsd:simpleType>

</xsd:element>

</xsd:schema>

// Results in the following mapping:

**module** http\_www\_example\_org\_union {

**import** **from** XSD **all**;

**type union** E21unnamed **{**

 XSD.Float alt\_,

 XSD.Integer alt\_1,

 XSD.String alt\_2,

**}**

**with** **{**
 **variant** "name as uncapitalized";

 **variant** "element";
 **variant** "useUnion";

 **variant**(alt\_, alt\_1, alt\_2) "name as ''";
}
}

**with** {

 **encode** "XML";

 **variant** "namespace as 'www.example.org/union'";

 **variant** "controlNamespace 'http://www.w3.org/2001/XMLSchema-instance' prefix 'xsi'";

}

// For instance, the below structure:

**template** E21unnamed t\_UnionUnnamedInt := { alt\_1 := 1 }

// will result in the following encoding:

<?xml version="1.0" encoding="UTF-8"?>
<e21unnamed xmlns='www.example.org/union' xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xsi:type='xsd:integer'>1</e21unnamed>

EXAMPLE 3: Mixed use of named and unnamed types:

<xsd:simpleType name="Time-or-int-or-boolean-or-dateRestricted">

 <xsd:union memberTypes="xsd:time e21memberlist">

 <xsd:simpleType>

 <xsd:restriction base="xsd:date"/>

 </xsd:simpleType>

 </xsd:union>

</xsd:simpleType>

//Will be mapped to the TTCN-3 type definition:

**type union** Time\_or\_int\_or\_boolean\_or\_dateRestricted **{**

 XSD.Time time,

 XSD.Integer integer\_,

 XSD.Boolean boolean\_,

 XSD.Date alt\_

}

**with {**

 **variant** "name as 'Time-or-int-or-boolean-or-dateRestricted'";

 **variant** "useUnion";

 **variant** (integer\_) "name as 'integer'";

 **variant** (boolean\_) "name as 'boolean'";

 **variant**(alt\_) "name as ''";

}

The only supported facet is *enumeration*, allowing mixing enumerations of different kinds.

EXAMPLE 4: Mapping member type with an enumeration facet:

<xsd:element name="maxOccurs">

 <xsd:simpleType>

 <xsd:union memberTypes="xsd:nonNegativeInteger">

 <xsd:simpleType>

 <xsd:restriction base="xsd:token">

 <xsd:enumeration value="unbounded"/>

 </xsd:restriction>

 </xsd:simpleType>

 </xsd:union>

 </xsd:simpleType>

</xsd:element>

//Will be translated to TTCN-3 as:

**type union** MaxOccurs {

 XSD.NonNegativeInteger nonNegativeInteger,

 **enumerated** **{**unbounded**}** alt\_

}

**with {**

 **variant** "name as uncapitalized";

 **variant** "element";

 **variant** "useUnion";

 **variant**(alt\_) "name as ''";

}

EXAMPLE 5: Mapping member types with enumeration facets applied to different member types:

<xsd:simpleType name="e22">

 <restriction base="e21unnamed">

 <xsd:enumeration value="20"/>

 <xsd:enumeration value="50"/>

 <xsd:enumeration value="small"/>

 </xsd:restriction>

</xsd:simpleType>

// will be translated to:

**type** E21unnamed E22 ({alt\_:=20.0},{alt\_:=50.0},{alt\_2:="small"})
**with** **{**
 **variant** "name as uncapitalized";

**}**