

XML Schema for WSML Logical Expressions

Table of Contents

- [Schema Document Properties](#)
- [Global Schema Components](#)
 - [Element: true](#)
 - [Element: false](#)
 - [Element: and](#)
 - [Element: or](#)
 - [Element: neg](#)
 - [Element: naf](#)
 - [Element: implies](#)
 - [Element: impliedBy](#)
 - [Element: equivalent](#)
 - [Element: constraint](#)
 - [Element: forall](#)
 - [Element: exists](#)
 - [Complex Type: quantifiedFormulaType](#)
 - [Complex Type: fullFormulaType](#)
 - [Complex Type: fullDualFormulaType](#)
 - [Complex Type: fullMultiFormulaType](#)
 - [Element: atom](#)
 - [Complex Type: termType](#)
 - [Complex Type: pathExpressionType](#)
 - [Element: molecule](#)

[top](#)

Schema Document Properties

Target Namespace	http://www.wsmo.org/2004/wsml
Element and Attribute Namespaces	<ul style="list-style-type: none">• Global element and attribute declarations belong to this schema's target namespace.• By default, local element declarations belong to this schema's target namespace.• By default, local attribute declarations have no namespace.
Schema Composition	<ul style="list-style-type: none">• This schema includes components from the following schema document(s):<ul style="list-style-type: none">◦ http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsml-identifiers
Documentation	version: \$Revision: 1.3 \$ date: \$Date: 2004/10/25 18:58:48 \$ author: Jos de Bruijn this schema is a module, which belongs to the WSML/XML schema specification. This schema provides WSML/XML syntax for logical expressions. This syntax for logical expressions is inspired by the RuleML FOL (First-Order Logic) variant. Necessary additions for WSML were the introduction of negation-as-failure, constraints, path expressions and the use of arbitrary terms as atom identifiers.

Declared Namespaces

Prefix	Namespace
Default namespace	http://www.wsmo.org/2004/wsm1
xml	http://www.w3.org/XML/1998/namespace
xs	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```
<xs:schema targetNamespace="http://www.wsmo.org/2004/wsm1"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:include
    schemaLocation="http://www.wsmo.org/2004/d16/d16.1/v0.2/xml-syntax/wsm1-ider
    ...
  </xs:schema>
```

[top](#)

Global Schema Components

Element: **true**

Name	true
Type	anyType
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	Represents universal truth (equivalent to a disjunction of a predicate and its complement)

XML Instance Representation

```
<true> ... </true>
```

Schema Component Representation

```
<xs:element name="true"/>
```

[top](#)

Element: **false**

Name	false
Type	anyType
<u>Nilable</u>	no
<u>Abstract</u>	no
Documentation	Represents universal falsehood (equivalent to a conjunction of a predicate and its complement)

XML Instance Representation

```
<false> ... </false>
```

Schema Component Representation

```
<xs:element name="false" />
```

[top](#)

Element: **and**

Name	and
Type	fullMultiFormulaType
Nilable	no
Abstract	no
Documentation	Conjunction

XML Instance Representation

```
<and>
  Start Choice [1..*]
  <atom> ... </atom> [1]
  <molecule> ... </molecule> [1]
  <and> ... </and> [1]
  <or> ... </or> [1]
  <neg> ... </neg> [1]
  <implies> ... </implies> [1]
  <impliedBy> ... </impliedBy> [1]
  <equivalent> ... </equivalent> [1]
  <forall> ... </forall> [1]
  <exists> ... </exists> [1]
  <constraint> ... </constraint> [1]
  <>true> ... </true> [1]
  <false> ... </false> [1]
  End Choice
</and>
```

Schema Component Representation

```
<xs:element name="and" type=" fullMultiFormulaType " />
```

[top](#)

Element: **or**

Name	or
Type	fullMultiFormulaType
Nilable	no
Abstract	no
Documentation	Disjunction

XML Instance Representation

```
<or>
  Start Choice [1..*]
  <atom> ... </atom> [1]
  <molecule> ... </molecule> [1]
  <and> ... </and> [1]
  <or> ... </or> [1]
  <neg> ... </neg> [1]
  <implies> ... </implies> [1]
```

```

    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</or>

```

Schema Component Representation

```
<xs:element name="or" type="fullMultiFormulaType" />
```

[top](#)

Element: **neg**

Name	neg
Type	fullFormulaType
Nilable	no
Abstract	no
Documentation	Classical negation

XML Instance Representation

```

<neg>
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</neg>

```

Schema Component Representation

```
<xs:element name="neg" type="fullFormulaType" />
```

[top](#)

Element: **naf**

Name	naf
Type	fullFormulaType
Nilable	no
Abstract	no
Documentation	Negation-as-failure

XML Instance Representation

```

<naf>
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <>true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</naf>

```

Schema Component Representation

```

<xs:element name="naf" type="fullFormulaType" />

```

[top](#)

Element: **implies**

Name	implies
Type	fullDualFormulaType
Nilable	no
Abstract	no
Documentation	Right implication (corresponds with necessary conditions). The first formula nested inside the 'implies' element, implies the second formula. For an implication, exactly two formulas must be nested inside the implication element.

XML Instance Representation

```

<implies>
  Start Choice [2..2]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]

```

```

    <true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</implies>

```

Schema Component Representation

```
<xs:element name="implies" type="fullDualFormulaType" />
```

[top](#)

Element: **impliedBy**

Name	impliedBy
Type	fullDualFormulaType
Nilable	no
Abstract	no
Documentation	Left implication (corresponds with sufficient conditions). The first formula nested inside the 'implies' element, is implied by the second formula. For an implication, exactly two formulas must be nested inside the implication element.

XML Instance Representation

```

<impliedBy>
  Start Choice [2..2]
  <atom> ... </atom> [1]
  <molecule> ... </molecule> [1]
  <and> ... </and> [1]
  <or> ... </or> [1]
  <neg> ... </neg> [1]
  <implies> ... </implies> [1]
  <impliedBy> ... </impliedBy> [1]
  <equivalent> ... </equivalent> [1]
  <forall> ... </forall> [1]
  <exists> ... </exists> [1]
  <constraint> ... </constraint> [1]
  <true> ... </true> [1]
  <false> ... </false> [1]
  End Choice
</impliedBy>

```

Schema Component Representation

```
<xs:element name="impliedBy" type="fullDualFormulaType" />
```

[top](#)

Element: **equivalent**

Name	equivalent
Type	fullDualFormulaType
Nilable	no
Abstract	no
Documentation	Dual implication (corresponds with necessary and sufficient conditions). The first formula nested inside the 'implies' element, is implied by the second formula. For an implication, exactly two formulas must be nested inside the implication element.

XML Instance Representation

```

<equivalent>
  Start Choice [2..2]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <>true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</equivalent>

```

Schema Component Representation

```

<xs:element name="equivalent" type="fullDualFormulaType" />

```

[top](#)

Element: **constraint**

Name	constraint
Type	fullFormulaType
Nilable	no
Abstract	no
Documentation	The formula nested inside this element is interpreted as an integrity constraint

XML Instance Representation

```

<constraint>
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
  End Choice
</constraint>

```

```

    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</constraint>

```

Schema Component Representation

```
<xs:element name="constraint" type="fullFormulaType" />
```

[top](#)

Element: forall

Name	forall
Type	quantifiedFormulaType
Nilable	no
Abstract	no
Documentation	Universal quantification.

XML Instance Representation

```

<forall>
  <var> wsmlVariable </var> [1..*]
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</forall>

```

Schema Component Representation

```
<xs:element name="forall" type="quantifiedFormulaType" />
```

[top](#)

Element: exists

Name	exists
Type	quantifiedFormulaType
Nilable	no
Abstract	no
Documentation	Existential quantification

XML Instance Representation

```
<exists>
  <var> wsmlVariable </var> [1..*]
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <>true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</exists>
```

Schema Component Representation

```
<xs:element name="exists" type=" quantifiedFormulaType "/>
```

[top](#)

Complex Type: **quantifiedFormulaType**

Super-types:	None
Sub-types:	None

Name	quantifiedFormulaType
Abstract	no
Documentation	For now the fullFormulaType is repeated after the 'var' element, because I don't know how to include it *after* 'var'.

XML Instance Representation

```
<...>
  <var> wsmlVariable </var> [1..*]
  Start Choice [1]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <>true> ... </true> [1]
    <false> ... </false> [1]
  End Choice
</...>
```

Schema Component Representation

```
<xs:complexType name="quantifiedFormulaType">
  <xs:sequence>
```

```

<xs:element name="var" type=" wsmlVariable "
maxOccurs="unbounded"/>
<xs:choice>
  <xs:element ref=" atom "/>
  <xs:element ref=" molecule "/>
  <xs:element ref=" and "/>
  <xs:element ref=" or "/>
  <xs:element ref=" neg "/>
  <xs:element ref=" implies "/>
  <xs:element ref=" impliedBy "/>
  <xs:element ref=" equivalent "/>
  <xs:element ref=" forall "/>
  <xs:element ref=" exists "/>
  <xs:element ref=" constraint "/>
  <xs:element ref=" true "/>
  <xs:element ref=" false "/>
</xs:choice>
</xs:sequence>
</xs:complexType>

```

[top](#)

Complex Type: **fullFormulaType**

Super-types:	None
Sub-types:	None

Name	fullFormulaType
Abstract	no
Documentation	The basic type for WSML-Full formulas, allowing exactly one formula

XML Instance Representation

```

<...>
  Start Choice [1]
  <atom> ... </atom> [1]
  <molecule> ... </molecule> [1]
  <and> ... </and> [1]
  <or> ... </or> [1]
  <neg> ... </neg> [1]
  <implies> ... </implies> [1]
  <impliedBy> ... </impliedBy> [1]
  <equivalent> ... </equivalent> [1]
  <forall> ... </forall> [1]
  <exists> ... </exists> [1]
  <constraint> ... </constraint> [1]
  <>true> ... </true> [1]
  <false> ... </false> [1]
  End Choice
</...>

```

Schema Component Representation

```

<xs:complexType name="fullFormulaType">
  <xs:choice>
    <xs:element ref=" atom "/>
    <xs:element ref=" molecule "/>
    <xs:element ref=" and "/>
    <xs:element ref=" or "/>

```

```

<xs:element ref=" neg "/>
<xs:element ref=" implies "/>
<xs:element ref=" impliedBy "/>
<xs:element ref=" equivalent "/>
<xs:element ref=" forall "/>
<xs:element ref=" exists "/>
<xs:element ref=" constraint "/>
<xs:element ref=" true "/>
<xs:element ref=" false "/>
</xs:choice>
</xs:complexType>

```

[top](#)

Complex Type: **fullDualFormulaType**

Super-types: None

Sub-types: None

Name	fullDualFormulaType
Abstract	no
Documentation	The basic type for WSML-Full formulas, allowing exactly two formulas (for implication)

XML Instance Representation

```

<...>
  Start Choice [2..2]
    <atom> ... </atom> [1]
    <molecule> ... </molecule> [1]
    <and> ... </and> [1]
    <or> ... </or> [1]
    <neg> ... </neg> [1]
    <implies> ... </implies> [1]
    <impliedBy> ... </impliedBy> [1]
    <equivalent> ... </equivalent> [1]
    <forall> ... </forall> [1]
    <exists> ... </exists> [1]
    <constraint> ... </constraint> [1]
    <>true> ... </true> [1]
    <>false> ... </false> [1]
  End Choice
</...>

```

Schema Component Representation

```

<xs:complexType name="fullDualFormulaType">
  <xs:choice minOccurs="2" maxOccurs="2">
    <xs:element ref=" atom "/>
    <xs:element ref=" molecule "/>
    <xs:element ref=" and "/>
    <xs:element ref=" or "/>
    <xs:element ref=" neg "/>
    <xs:element ref=" implies "/>
    <xs:element ref=" impliedBy "/>
    <xs:element ref=" equivalent "/>
    <xs:element ref=" forall "/>
    <xs:element ref=" exists "/>
    <xs:element ref=" constraint "/>
    <xs:element ref=" true "/>
  </xs:choice>
</xs:complexType>

```

```
<xs:element ref=" false "/>
</xs:choice>
</xs:complexType>
```

[top](#)

Complex Type: **fullMultiFormulaType**

Super-types: None

Sub-types: None

Name	fullMultiFormulaType
Abstract	no
Documentation	The basic type for WSML-Full formulas, allowing arbitrarily many formulas, for conjunctions and disjunctions

XML Instance Representation

```
<...>
  Start Choice [1..*]
  <atom> ... </atom> [1]
  <molecule> ... </molecule> [1]
  <and> ... </and> [1]
  <or> ... </or> [1]
  <neg> ... </neg> [1]
  <implies> ... </implies> [1]
  <impliedBy> ... </impliedBy> [1]
  <equivalent> ... </equivalent> [1]
  <forall> ... </forall> [1]
  <exists> ... </exists> [1]
  <constraint> ... </constraint> [1]
  <>true> ... </true> [1]
  <>false> ... </false> [1]
  End Choice
</...>
```

Schema Component Representation

```
<xs:complexType name="fullMultiFormulaType">
  <xs:choice maxOccurs="unbounded">
    <xs:element ref=" atom "/>
    <xs:element ref=" molecule "/>
    <xs:element ref=" and "/>
    <xs:element ref=" or "/>
    <xs:element ref=" neg "/>
    <xs:element ref=" implies "/>
    <xs:element ref=" impliedBy "/>
    <xs:element ref=" equivalent "/>
    <xs:element ref=" forall "/>
    <xs:element ref=" exists "/>
    <xs:element ref=" constraint "/>
    <xs:element ref=" true "/>
    <xs:element ref=" false "/>
  </xs:choice>
</xs:complexType>
```

[top](#)

Element: **atom**

Name	atom
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	An atom denotes a relation with an n-ary domain, where n is the arity of the predicate

XML Instance Representation

```

<atom
  negated=" xs:boolean [0..1]">
  Start Choice [1]
    <name> wsmlAnyID </name> [1]
    <term> termType </term> [1]
  End Choice
  Start Choice [0..*]
    <argument> pathExpressionType </argument> [1]
    <argumentID> wsmlAnyID </argumentID> [1]
  End Choice
</atom>

```

Schema Component Representation

```

<xs:element name="atom">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element name="name" type=" wsmlAnyID "/>
        <xs:element name="term" type=" termType "/>
      </xs:choice>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="argument" type=" pathExpressionType "/>
        <xs:element name="argumentID" type=" wsmlAnyID "/>
      </xs:choice>
    </xs:sequence>
    <xs:attribute name="negated" type=" xs:boolean "
      default="false"/>
  </xs:complexType>
</xs:element>

```

[top](#)

Complex Type: **termType**

<i>Super-types:</i>	None
<i>Sub-types:</i>	<ul style="list-style-type: none"> • pathExpressionType (by extension)

Name	termType
Abstract	no
Documentation	A term can be a wsmlAnyID (URI, literal, variable or anonymous ID) or a constructed term (corresponding to a function symbol), where an argument can be any path expression.

XML Instance Representation

```

<...>
  <name> wsmlAnyID </name> [1]
  <argument> pathExpressionType </argument> [0..*]
</...>

```

Schema Component Representation

```

<xs:complexType name="termType">
  <xs:sequence>
    <xs:element name="name" type=" wsmlAnyID "/>
    <xs:element name="argument" type=" pathExpressionType "
      minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

```

[top](#)

Complex Type: **pathExpressionType**

Super-types: [termType](#) < **pathExpressionType** (by extension)

Sub-types: None

Name	pathExpressionType
Abstract	no
Documentation	A path expression allows indicating a nesting of terms. A path expression of the form 'a.b' represents the value for attribute 'b' for object 'a'.

XML Instance Representation

```

<...>
  <name> wsmlAnyID </name> [1]
  <argument> pathExpressionType </argument> [0..*]
  Start Sequence [0..*]
    Start Choice [1]
      <nestingID> wsmlAnyID </nestingID> [1]
      <nestingTerm> termType </nestingTerm> [1]
    End Choice
  End Sequence
</...>

```

Schema Component Representation

```

<xs:complexType name="pathExpressionType">
  <xs:complexContent>
    <xs:extension base=" termType ">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
          <xs:element name="nestingID" type=" wsmlAnyID "/>
          <xs:element name="nestingTerm" type=" termType "/>
        </xs:choice>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

[top](#)

Element: **molecule**

Name	molecule
Type	Locally-defined complex type
Nilable	no
Abstract	no
Documentation	A molecule denotes either a memberOf, a subConceptOf, an attributeOf or an attributeValueOf relationship. Or a combination of those

XML Instance Representation

```

<molecule>
  Start Choice [1]
    <name> wsmlAnyID </name> [1]
    <term> termType </term> [1]
  End Choice
  <attributeDefinition>
    type=" xs:string (value comes from list:
    {'inferring'|'constraining'}) [0..1]"> [0..*]
    <name> wsmlAnyID </name> [1]
    <type> wsmlAnyID </type> [1]
  </attributeDefinition>
  <attributeValueDefinition> [0..*]
    <name> wsmlAnyID </name> [1]
    <value> wsmlAnyID </value> [1..*]
  </attributeValueDefinition>
  <isa>
    type=" xs:string (value comes from list:
    {'memberOf'|'subConceptOf'}) [0..1]"> [0..1]
    xs:string
  </isa>
</molecule>

```

Schema Component Representation

```

<xs:element name="molecule">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element name="name" type=" wsmlAnyID "/>
        <xs:element name="term" type=" termType "/>
      </xs:choice>
      <xs:element name="attributeDefinition" minOccurs="0"
      maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type=" wsmlAnyID "/>
            <xs:element name="type" type=" wsmlAnyID "/>
          </xs:sequence>
          <xs:attribute name="type" default="constraining">
            <xs:simpleType>
              <xs:restriction base=" xs:string ">
                <xs:enumeration value="inferring"/>
                <xs:enumeration value="constraining"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:complexType>
      </xs:element>
      <xs:element name="attributeValueDefinition" minOccurs="0"
      maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type=" wsmlAnyID "/>

```

```
        <xs:element name="value" type=" wsmlAnyID "
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="isa" minOccurs="0">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base=" xs:string ">
          <xs:attribute name="type">
            <xs:simpleType>
              <xs:restriction base=" xs:string ">
                <xs:enumeration value="memberOf" />
                <xs:enumeration value="subConceptOf" />
              </xs:restriction>
            </xs:simpleType>
          </xs:attribute>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```

[top](#)

Generated by [xs3p](#).