«disjunct»

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
finiteType

0 «exhaustive set» 0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType

{ cardinality = finite }

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType

0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
ordinalType

0 «exhaustive set» 0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
partiallyOrderedType

{ order = partial }

{ order = total }

not bounded, though some sub-classes are bounded

{ cardinality countably infinite }

{ numeric = false }

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonNumericType

0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonOrderedType

0

{ bounded = true }

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedType

0 «exhaustive set» 0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType

{ bounded = false }

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
durationOrTimeCoordinate

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
literalType
length
maxLength
minLength

«disjoint»

0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
numericType
maxExclusive
minExclusive
maxInclusive
minInclusive

{ numeric = true }
{ Some partially ordered subtypes, some totally ordered aubtypes. For presentational purposes, all subtypes may have a quality of ordering, in some regards }

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
floatingPointType

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
decimal
fractionDigits
totalDigits

<<constructedType>>
http://www.w3.org/2001/XMLSchema
integer

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedInteger

<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonNegativeInteger

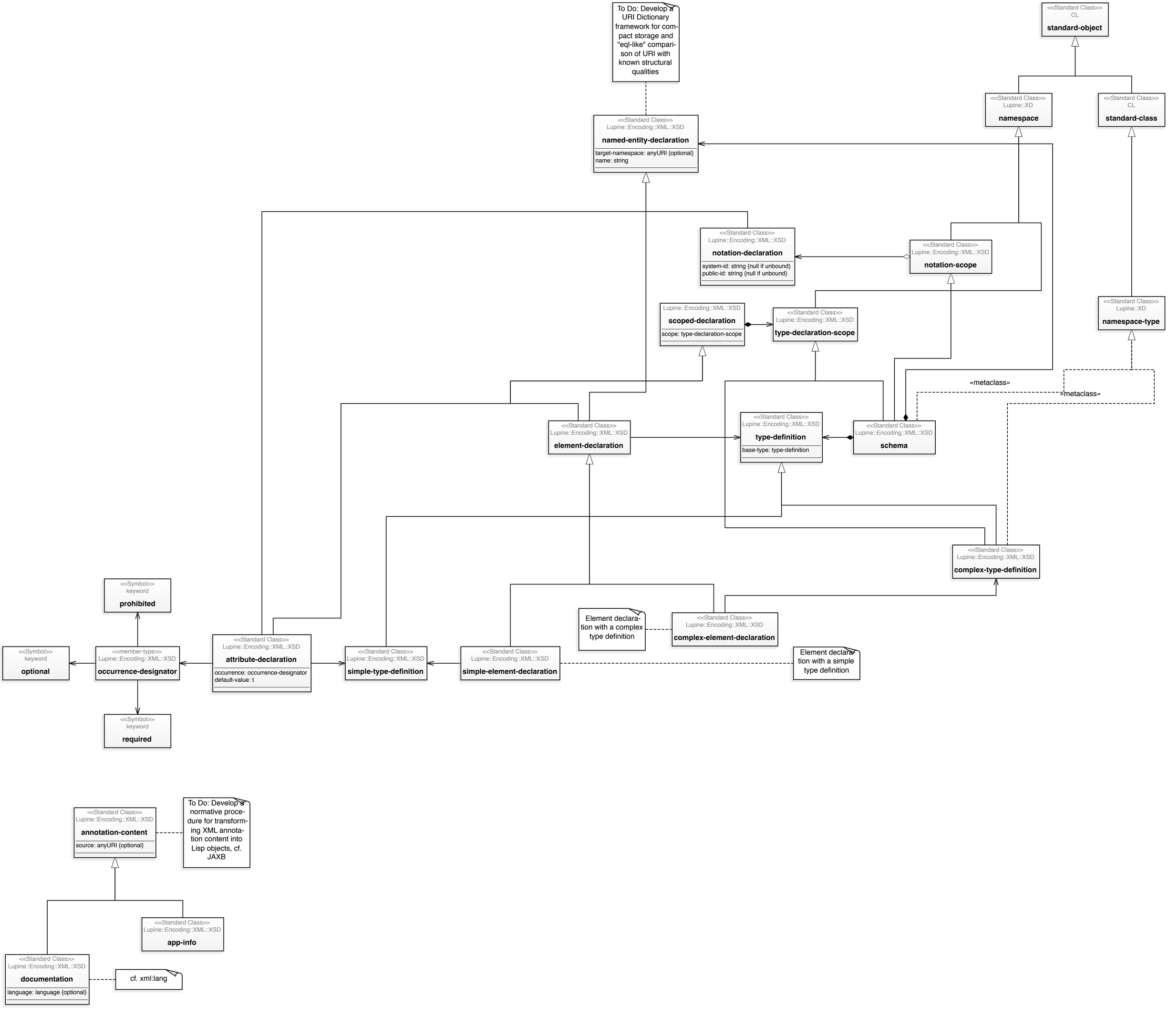
«exhaustive set» 0

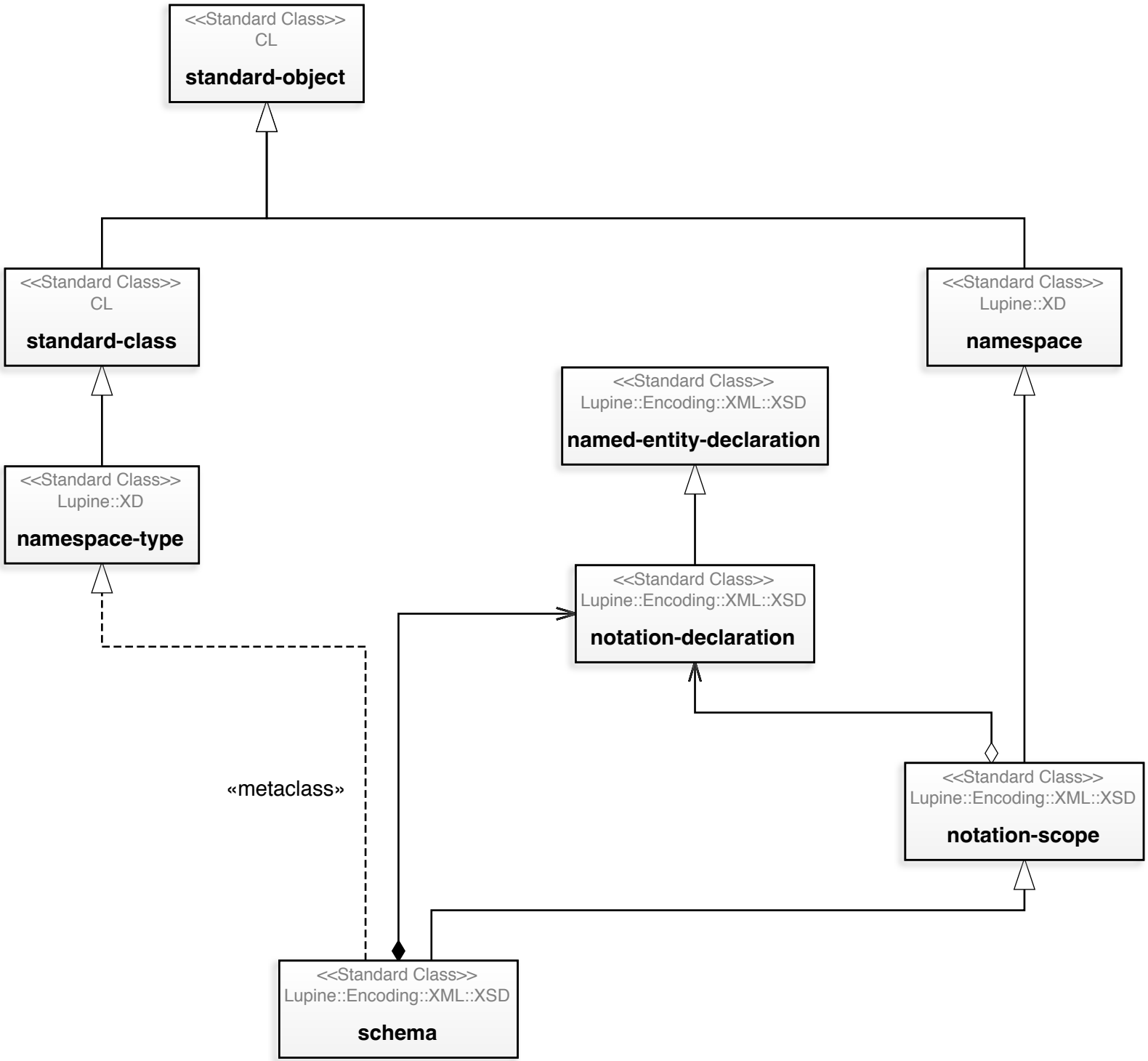
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
signedBoundedInteger

0

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
unsignedBoundedInteger

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType
values: total, partial, false
Lupine::Encoding::XML::XSD::Implementation::Lisp
URI
values: finite, countably infinite
<<metaclass>>
Lupine::Encoding::XML::XSD::Implementation
typeClass
cardinality: cardinalityType
bounded: boolean
ordered: orderedType
whitespace: whitespaceType
numeric: boolean {defaultValue = false}
<<enumeration>>
Lupine::Encoding::XML::XSD::Implementation
cardinalityType
<<enumeration>>
Lupine::Encoding::XML::XSD::Implementation
whitespaceType
values: preserve, collapse, replace
nb (Presentation) Cubetto uses an arbitrary ordering in the attribute list, such that the user is unable to edit
Note that each XML element within an XML document could be considered a "namespace registry" individually - each element servimg as a context for namespace name/URI binding. Comsult DOM for how this is handled within XML DOM (Desigm question: Will the XSD implementation extend on the DOM and/or SAX model in CXML?)
Lupine::Encoding::XML::XSD::Implementation::Lisp
namespace-registry
get-namespace
name registry
set-namespace
ns name registry
baseType
To Do: Design decision, should an xml-namespace serve as a reference to any schema defined as targeting the namespace?
*
Lupine::Encoding::XML::XSD::Implementation::Lisp
xml-namespace
uri
namespace-uri
«instance-of»
0..1
<<stack-object>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
null-uri
«instance-of»
namespace-uri
Note that an OWL ontology defining XSD datatypes may be denoted as containing an implicit schema part, identified by the containing ontology
<<Standard Class>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
xml-schema
<<stack-object>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
null-namespace
0..1
type-target-namespace
type-schema
<<Standard Class>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
xsd-type-class
whitespace-handling
bounded-p: boolean {initform: nil}
ordered-p
lisp-type
schema: xml-schema {null if unbound}
finite-p: boolean
numeric-p: boolean {initform: nil}
target-namespace: xml-namespace {initform: null-namespace}
<<metaclass>>
Lupine::Encoding::XML::XSD::Implementation
primitiveTypeClass
<<metaclass>>
Lupine::Encoding::XML::XSD::Implementation
constructedTypeClass
baseType: typeClass
<<Standard Class>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
xsd-primitive-type-class
<<Standard Class>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
xsd-constructed-type-class
base-type: xsd-type-class
Lupine::Encoding::XML::XSD::Implementation
timeCoordinateTypeClass
explicitTimezone {defaultValue = optional}
<<Standard Class>>
Lupine::Encoding::XML::XSD::Implementation::Lisp
xsd-time-coordinate-type-class
timezone-optional-p: boolean {defaultValue = T}
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType


To Do: Develop a URI Dictionary framework for compact storage and "eql-like" comparison of URI with known structural qualities
<<Standard Class>>
CL
standard-object
<<Standard Class>>
Lupine::XD
namespace
<<Standard Class>>
CL
standard-class
<<Standard Class>>
Lupine::Encoding::XML::XSD
named-entity-declaration
target-namespace: anyURI {optional}
name: string
<<Standard Class>>
Lupine::Encoding::XML::XSD
notation-declaration
system-id: string {null if unbound}
public-id: string {null if unbound}
<<Standard Class>>
Lupine::Encoding::XML::XSD
notation-scope
Lupine::Encoding::XML::XSD
scoped-declaration
scope: type-declaration-scope
<<Standard Class>>
Lupine::Encoding::XML::XSD
type-declaration-scope
<<Standard Class>>
Lupine::XD
namespace-type
«metaclass»
«metaclass»
<<Standard Class>>
Lupine::Encoding::XML::XSD
element-declaration
<<Standard Class>>
Lupine::Encoding::XML::XSD
type-definition
base-type: type-definition
<<Standard Class>>
Lupine::Encoding::XML::XSD
schema
<<Standard Class>>
Lupine::Encoding::XML::XSD
complex-type-definition
<<Symbol>>
keyword
prohibited
<<Symbol>>
keyword
optional
<<member-type>>
Lupine::Encoding::XML::XSD
occurrence-designator
<<Standard Class>>
Lupine::Encoding::XML::XSD
attribute-declaration
occurrence: occurrence-designator
default-value: t
<<Standard Class>>
Lupine::Encoding::XML::XSD
simple-type-definition
<<Standard Class>>
Lupine::Encoding::XML::XSD
simple-element-declaration
Element declaration with a complex type definition
<<Standard Class>>
Lupine::Encoding::XML::XSD
complex-element-declaration
Element declaration with a simple type definition
<<Symbol>>
keyword
required
<<Standard Class>>
Lupine::Encoding::XML::XSD
annotation-content
source: anyURI {optional}
To Do: Develop a normative procedure for transforming XML annotation content into Lisp objects, cf. JAXB
<<Standard Class>>
Lupine::Encoding::XML::XSD
app-info
<<Standard Class>>
Lupine::Encoding::XML::XSD
documentation
language: language {optional}
cf. xml:lang


<<Standard Class>>
CL
standard-object
<<Standard Class>>
CL
standard-class
<<Standard Class>>
Lupine::XD
namespace
<<Standard Class>>
Lupine::Encoding::XML::XSD
named-entity-declaration
<<Standard Class>>
Lupine::XD
namespace-type
<<Standard Class>>
Lupine::Encoding::XML::XSD
notation-declaration
«metaclass»
<<Standard Class>>
Lupine::Encoding::XML::XSD
notation-scope
<<Standard Class>>
Lupine::Encoding::XML::XSD
schema

«return value»

Lupine::XD
ensure-instance
namespace
name

«method generic function»

«method generic function»

<<Standard Method>>
ensure-instance (String Uri-Dictionary)

<<Standard Method>>
ensure-instance (String URI-Registry)

Errs if URI schema not supported (?) or eather, would return a default "no compression" handler

<<Standard Class>>
Lupine::XD
namespace-type

type of namespace (abstract)

<<Standard Class>>
Lupine::XD
namespace

«specializer»

«calls»

«metaclass»

«metaclass»

URI registry effective contents

<<Standard Class>>
Lupine::URI
uri-registry

Lupine::URI
string-dictionary
equality-predicate: function

Fundamental use case: Return an allocated char= (?) equivalent of an input string, if string is already allocated in dictionary, such as may then be used for straightforward 'eq' comparison, else assume that string has not yet been allocated in dictionary, i.e is "new" and will not be EQ to any objects deriving from the contents of the dictionary. (Similar to semantics of CL symbols, but in a disjunct sense of package/namespace. May use string hash codes ,alternate to char-by-char comparison)

cializer»

«specializer»

<<Standard Method>>
get-schema-dictionary (string uri-registry)

<<Standard Class>>
Lupine::URI
uri
scheme-name: string

string represents URI scheme name

URI dictionary effective contents

global st… nary for URI (?)

<<Standard Class>>
Lupine::URI
uri-dictionary

assigned scheme

<<Standard Class>>
Lupine::URI
uri-scheme
scheme-name: string

To do: Stack allocation of common 'scheme name' values (http, https, file, urn) and 'namespace name' values (isbn, etc)

<<Standard Class>>
Lupine::URI
urn-namespace
namepace-name: string

assigned namespace

<<Standard Class>>
Lupine::URI
file-uri-dictionary

<<Standard Class>>
Lupine::URI
flat-uri-dictionary

scheme = urn

<<Standard Class>>
Lupine::URI
urn-registry

<<Standard Class>>
Lupine::URI
urn-namespace-dictionary

URN dictionary effective contents

<<Standard Class>>
Lupine::URI
urn
namepace-name: string

scheme = file

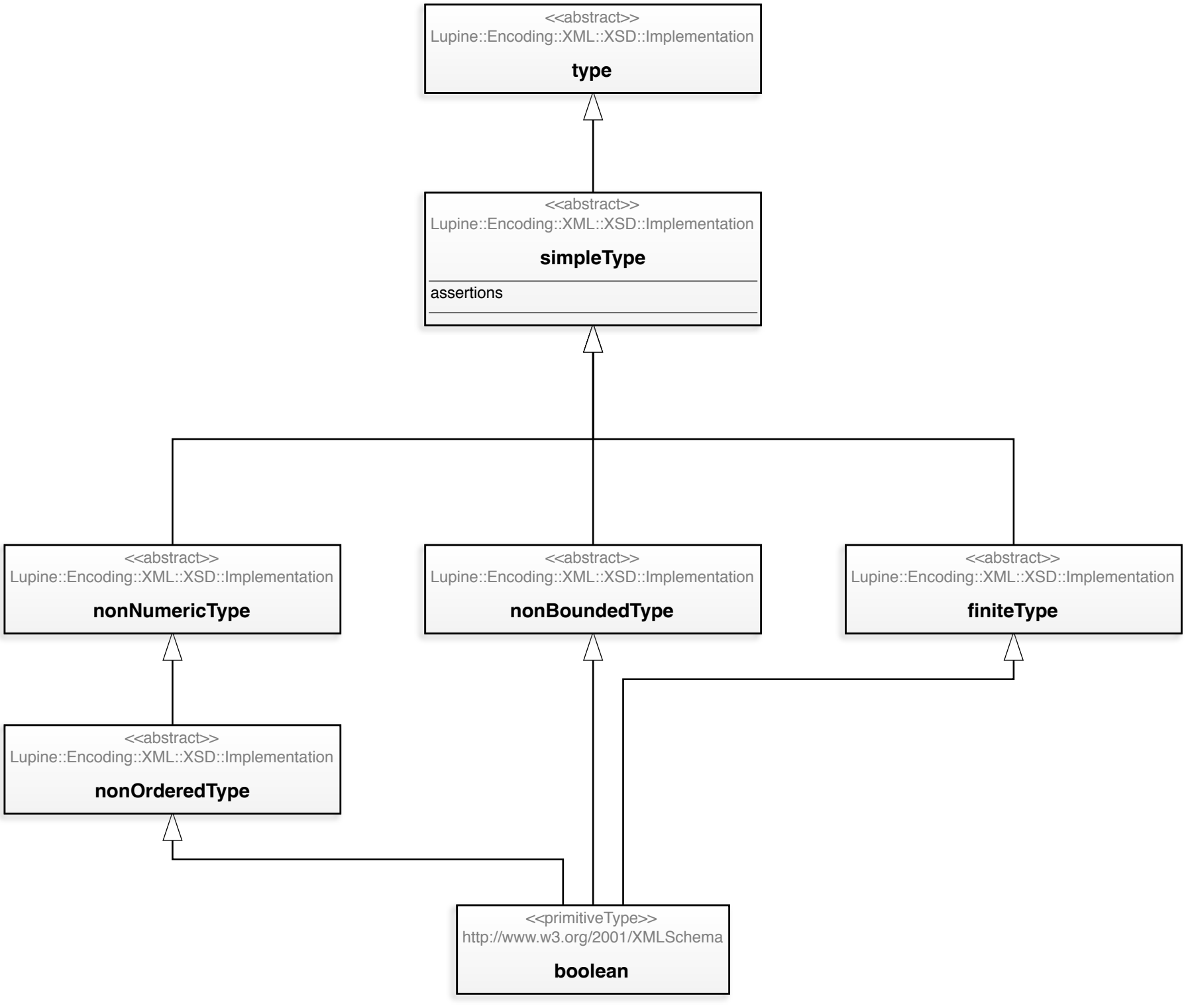
default dictionary for unrecognized URI schemes

<<Standard Class>>
Lupine::URI
http-uri-dictionary

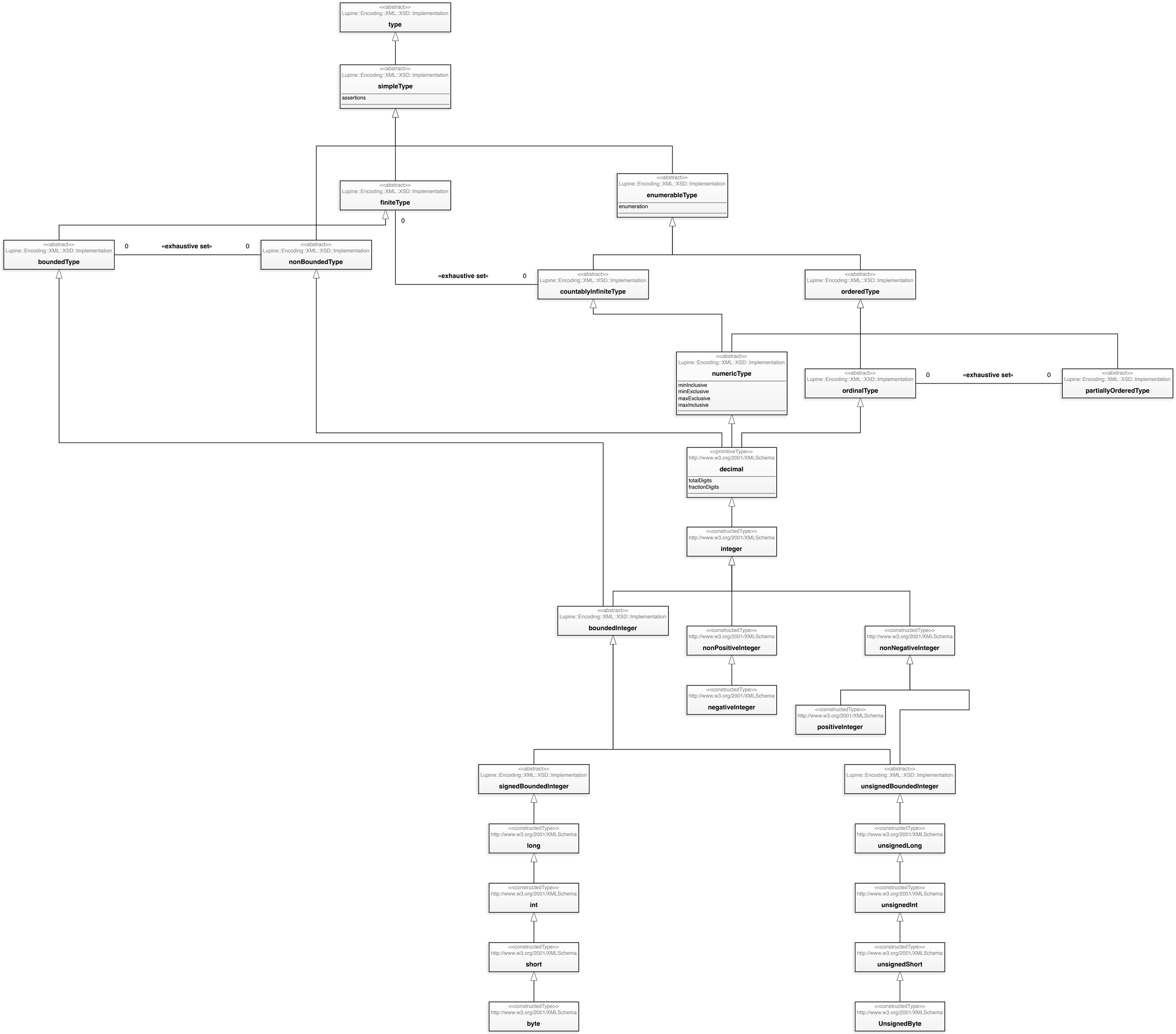
scheme = http

scheme = https

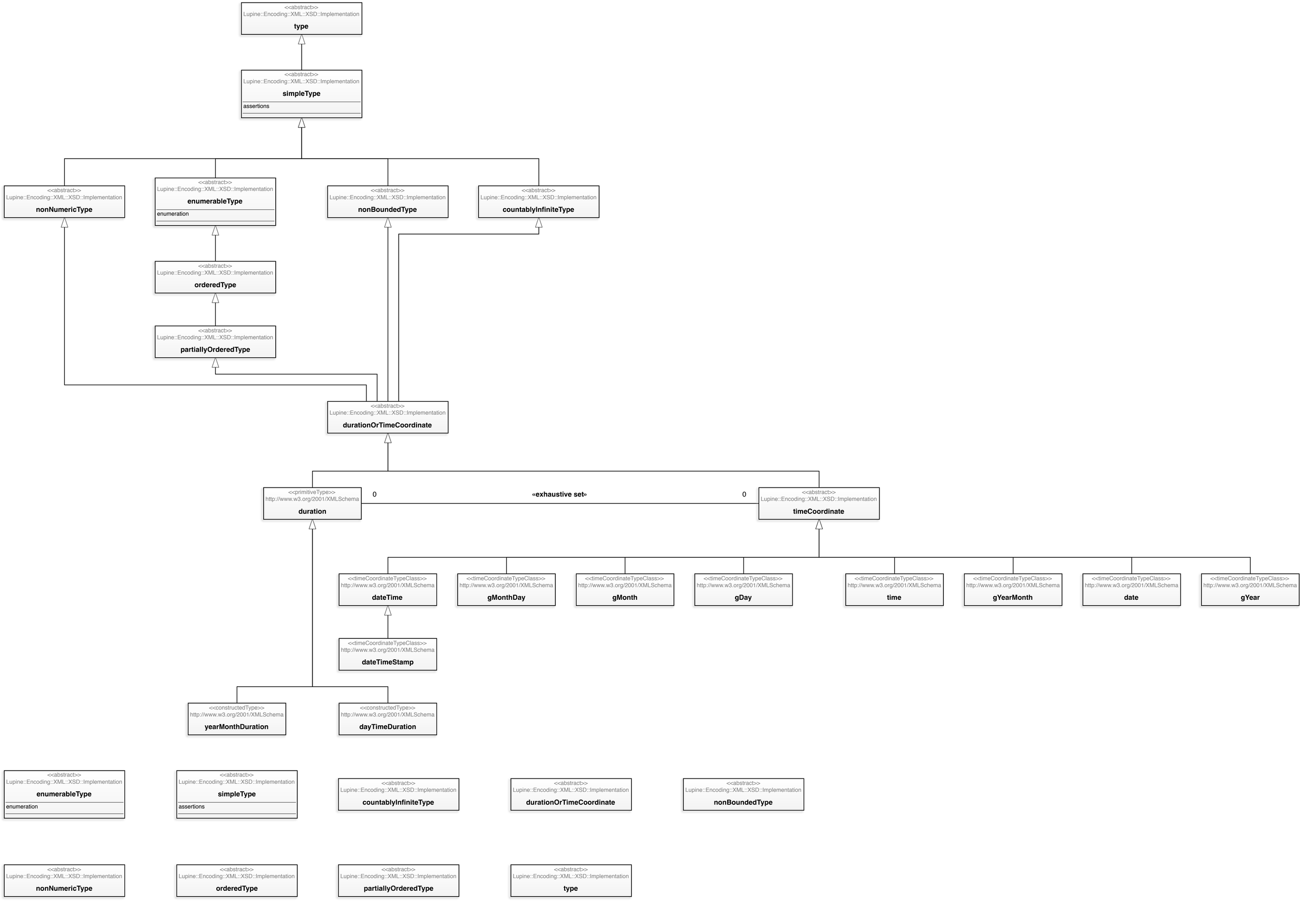
<<Standard Class>>
Lupine::URI
https-uri-dictionary

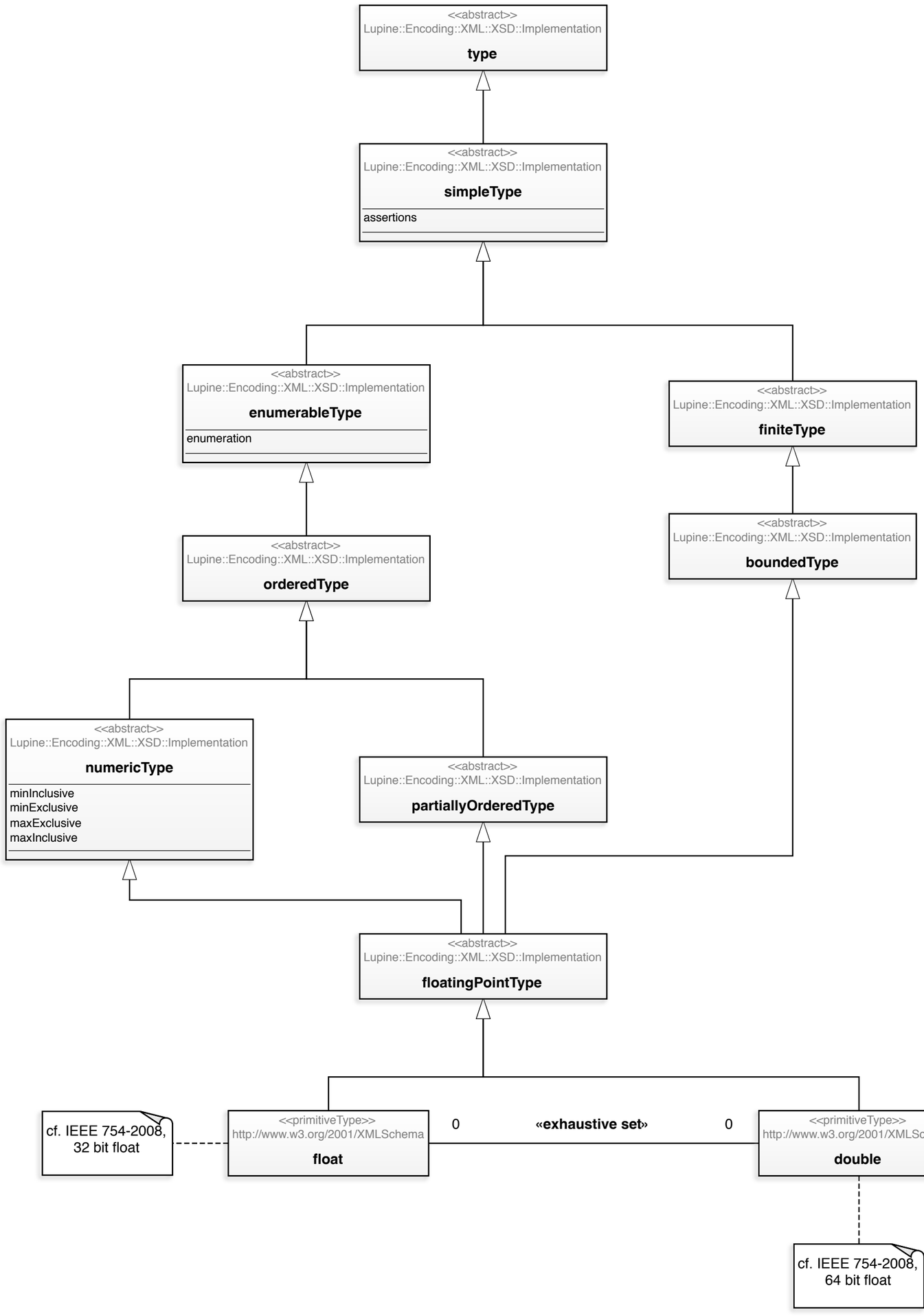

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonNumericType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
finiteType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonOrderedType
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
boolean

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
anySimpleType
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
anyAtomicType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
boolean
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
literalType
minLength
length
maxLength
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
numericType
minInclusive
minExclusive
maxExclusive
maxInclusive
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
timeCoordinate
Note that all sub-classes of this class reify timeCoordinateTypeClass (cf. the corresponding implementation metamodel)
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
duration
<<constructedType>>
http://www.w3.org/2001/XMLSchema
dayTimeDuration
<<constructedType>>
http://www.w3.org/2001/XMLSchema
yearMonthDuration
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
dateTime
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gYear
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gDay
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gMonthDay
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gYearMonth
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
date
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
time
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gMonth
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
dateTimeStamp
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
decimal
totalDigits
fractionDigits
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
floatingPointType
<<constructedType>>
http://www.w3.org/2001/XMLSchema
integer
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
float
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
double
<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonPositiveInteger
<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonNegativeInteger
<<constructedType>>
http://www.w3.org/2001/XMLSchema
long
<<constructedType>>
http://www.w3.org/2001/XMLSchema
negativeInteger
<<constructedType>>
http://www.w3.org/2001/XMLSchema
int
<<constructedType>>
http://www.w3.org/2001/XMLSchema
short
<<constructedType>>
http://www.w3.org/2001/XMLSchema
byte
<<constructedType>>
http://www.w3.org/2001/XMLSchema
unsignedLong
<<constructedType>>
http://www.w3.org/2001/XMLSchema
positiveInteger
<<constructedType>>
http://www.w3.org/2001/XMLSchema
unsignedInt
<<constructedType>>
http://www.w3.org/2001/XMLSchema
unsignedShort
<<constructedType>>
http://www.w3.org/2001/XMLSchema
UnsignedByte
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
string
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
lexicalLiteralType
<<constructedType>>
http://www.w3.org/2001/XMLSchema
normalizedString
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
base64Binary
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
NOTATION
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
hexBinary
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
QName
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
anyURI
<<constructedType>>
http://www.w3.org/2001/XMLSchema
token
<<constructedType>>
http://www.w3.org/2001/XMLSchema
language
<<constructedType>>
http://www.w3.org/2001/XMLSchema
Name
<<constructedType>>
http://www.w3.org/2001/XMLSchema
NMTOKEN
<<constructedType>>
http://www.w3.org/2001/XMLSchema
NCName
<<constructedType>>
http://www.w3.org/2001/XMLSchema
NMTOKENS
<<constructedType>>
http://www.w3.org/2001/XMLSchema
ID
<<constructedType>>
http://www.w3.org/2001/XMLSchema
IDREF
<<constructedType>>
http://www.w3.org/2001/XMLSchema
ENTITY
<<constructedType>>
http://www.w3.org/2001/XMLSchema
IDREFS
<<constructedType>>
http://www.w3.org/2001/XMLSchema
ENTITIES
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
numericType
maxExclusive
minExclusive
maxInclusive
minInclusive
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
literalType
length
maxLength
minLength
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
decimal
fractionDigits
totalDigits
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
floatingPointType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<constructedType>>
http://www.w3.org/2001/XMLSchema
integer
<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonNegativeInteger



<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
numericType
maxExclusive
minExclusive
maxInclusive
minInclusive
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
decimal
fractionDigits
totalDigits
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedInteger
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
finiteType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
ordinalType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
partiallyOrderedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
signedBoundedInteger
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
unsignedBoundedInteger
<<constructedType>>
http://www.w3.org/2001/XMLSchema
integer
<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonNegativeInteger


<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonNumericType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
partiallyOrderedType
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
durationOrTimeCoordinate
<<primitiveType>>
http://www.w3.org/2001/XMLSchema
duration
0
«exhaustive set»
0
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
timeCoordinate
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
dateTime
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gMonthDay
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gMonth
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gDay
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
time
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gYearMonth
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
date
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
gYear
<<timeCoordinateTypeClass>>
http://www.w3.org/2001/XMLSchema
dateTimeStamp
<<constructedType>>
http://www.w3.org/2001/XMLSchema
yearMonthDuration
<<constructedType>>
http://www.w3.org/2001/XMLSchema
dayTimeDuration



<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
numericType
maxExclusive
minExclusive
maxInclusive
minInclusive

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
finiteType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
floatingPointType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
partiallyOrderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonNumericType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonOrderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
literalType
minLength
length
maxLength

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
string

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
lexicalLiteralType

<<constructedType>>
http://www.w3.org/2001/XMLSchema
normalizedString

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
base64Binary

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
NOTATION

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
anyURI

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
QName

<<primitiveType>>
http://www.w3.org/2001/XMLSchema
hexBinary

<<constructedType>>
http://www.w3.org/2001/XMLSchema
token

<<constructedType>>
http://www.w3.org/2001/XMLSchema
language

<<constructedType>>
http://www.w3.org/2001/XMLSchema
Name

<<constructedType>>
http://www.w3.org/2001/XMLSchema
NMTOKEN

<<constructedType>>
http://www.w3.org/2001/XMLSchema
NCName

<<constructedType>>
http://www.w3.org/2001/XMLSchema
NMTOKENS

<<constructedType>>
http://www.w3.org/2001/XMLSchema
ID

<<constructedType>>
http://www.w3.org/2001/XMLSchema
ENTITY

<<constructedType>>
http://www.w3.org/2001/XMLSchema
IDREF

<<constructedType>>
http://www.w3.org/2001/XMLSchema
ENTITIES

<<constructedType>>
http://www.w3.org/2001/XMLSchema
IDREFS

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
literalType
length
maxLength
minLength

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions

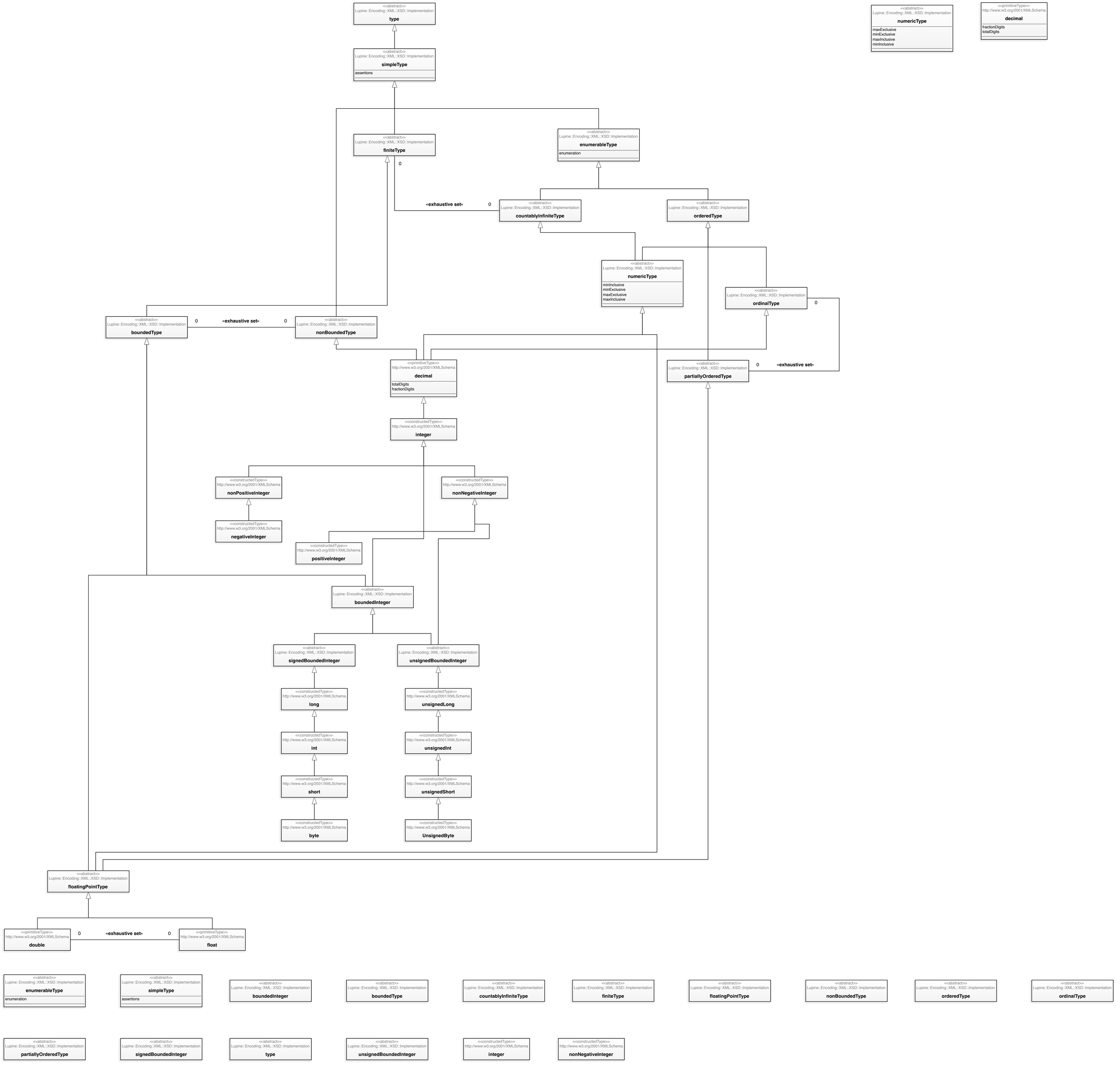
<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonNumericType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonOrderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type



<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
enumerableType
enumeration

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
simpleType
assertions

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedInteger

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
boundedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
countablyInfiniteType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
finiteType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
floatingPointType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
nonBoundedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
orderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
ordinalType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
partiallyOrderedType

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
signedBoundedInteger

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
type

<<abstract>>
Lupine::Encoding::XML::XSD::Implementation
unsignedBoundedInteger

<<constructedType>>
http://www.w3.org/2001/XMLSchema
integer

<<constructedType>>
http://www.w3.org/2001/XMLSchema
nonNegativeInteger