Content Models for RuleML

Tshering Dema, David Hirtle, Omair Shafiq, Derek Smith

2010-08-19, version 1.0

Introduction

This document is a collection of content models, i.e. the content permitted within a particular tag, for all RuleML tags as of version 1.0, organized alphabetically by module name. Each module is a grouping of related (XML) elements and/or attributes (prefixed with "@"). The content models are given in BNF-like DTD syntax. See http://www.ruleml.org/1.0/xsd/modules for the actual XML schemas of the modules and the RuleML glossary for the meaning of each tag.

Since RuleML is a family of sublanguages, it is important to note that the content model of a given tag often varies according to the current sublanguage. In such cases, all variations of the content model are provided along with the corresponding sublanguage(s). The modularization of RuleML, including all sublanguages, is explained at http://www.ruleml.org/modularization.

Content models may also vary depending on context, i.e. surrounding elements (especially parent elements). In these cases, the content models are listed under a heading such as "within x…" where x indicates the context.

For clarification on any RuleML-related topic, including this document, the RuleML-all mailing list may be quite helpful. The RuleML tutorial serves as an introduction.

Index

Atom

Atom

```
(context sensitive; see also the Holog module)

attributes: @closure

in datalog, nafdatalog, nafnegdatalog, negdatalog:
(
 (oid)?, degree?, (op | Rel), (slot)*,
 ( (arg|Ind|Data|Skolem|Var|Reify)+, (slot)* )?
)

in bindatalog:
(
 (oid)?, degree?, (op | Rel), (slot)*,
        ( (arg|Ind|Data|Skolem|Var|Reify), (arg|Ind|Data|Skolem|Var|Reify), (slot)* )?
)

in bindatagroundlog and bindatagroundfact:
(
        (oid)?, (op | Rel), (slot)*,
        ( (arg|Ind|Data|Skolem|Reify), (arg|Ind|Data|Skolem|Reify), (slot)* )?
        )

in hornlog & up (except framehohornlogeq):
(
        (oid)?, (op | Rel), (slot)*,
                ( (arg|Ind|Data|Skolem|Var|Reify|Expr|Plex)+, (slot)* )?
)
```

degree

```
in all sublanguages: (Data)
```

op

```
(context sensitive; see also the Holog, Equality and Expr modules)

within Atom...

in all sublanguages: ( Rel )
```

Rel

```
attributes: @iri

in all sublanguages: ( #PCDATA )
```

Connective

if

```
in datalog & down and hornlog, dishornlog, and hohornlog: (Atom | And | Or)

in negdatalog: (Atom | And | Or | Neg)

in nafdatalog & nafhornlog: (Atom | And | Or | Naf)

in nafnegdatalog: (Atom | And | Or | Neg | Naf)

in hornlogeq: (Atom | And | Or | Equal)

in hohornlogeq: (Uniterm | And | Or | | Neg | Naf | Equal)

in framehohornlogeq: (Atom | Uniterm | InstanceOf | SubclassOf | Signature | And | Or | Neg | Naf
| Equal)

in folog: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

in naffolog: (Atom | And | Or | Neg | Naf | Implies | Equivalent | Forall | Exists )

in fologeq: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal )

in naffologeq: (Atom | And | Or | Neg | Naf | Implies | Equivalent | Forall | Exists | Equal )
```

Implies

```
attributes: @closure, @direction, @material ( + @mapDirection and @mapClosure in folog & up)

in datalog & down and hornlog:
( oid?, ( then, if) | ( if, then) | ( (Atom | Rulebase | And | Or), Atom ) )

in negdatalog: ( oid?, ( then, if) | ( if, then) | ( (Atom | Rulebase | And | Or | Neg), (Atom |
Neg) ) )

in nafdatalog & nafhornlog: ( oid?, ( then, if) | ( if, then) | ( (Atom | And | Or | Naf), Atom )
)

in nafnegdatalog: (oid?, ( then, if) | ( if, then) | ( (Atom | And | Or | Neg | Naf), (Atom |
Neg) ))

in hornlogeq: ( oid?, ( then, if) | ( if, then) | ( (Atom | And | Or | Equal), (Atom | Equal) ))

in hohornlog: ( oid?, ( then, if) | ( if, then) | ( (Uniterm | And | Or | Neg | Naf), (Uniterm |
Neg) ) )

in hohornlogeq: ( oid?, ( then, if) | ( if, then) | ( ( Uniterm| And| Or| Neg| Naf| Equal),
(  Uniterm| Neg| Equal) ) )

in framehohornlogeq:
(
   oid?, ( then, if ) | ( if, then ) |
        (
               (Atom|Uniterm|InstanceOf|SubclassOf|Signature|And|Or| Neg|Naf|Equal),
               (Atom|Uniterm|InstanceOf|SubclassOf|Signature Neg|Naf|Equal)
         )
)

in dishornlog: ( oid?, ( then, if) | ( if, then) | ( (Atom | And | Or), (Atom | Or)))

in folog:
(
   oid?, (then, if) | (if, then) |
        (
         (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists ),
         (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )
```

```
        )
    )

    in naffolog:
    (
       oid?, (then, if) | (if, then) |
        (
            (Atom | And | Or | Neg | Naf | Implies | Equivalent | Forall | Exists ),
           (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )
         )
    )


    in fologeq:

    (
       oid?, (then, if) | (if, then) |
       (
         (Atom | And|Or | Neg | Implies | Equivalent | Forall | Exists | Equal ),
         (Atom | And|Or | Neg | Implies | Equivalent | Forall | Exists | Equal )                    )
)

    in naffologeq:
    (
       oid?, (then, if) | (if, then) |
         (
           (Atom | And|Or | Neg|Naf | Implies|Equivalent | Forall|Exists |Equal ),
           (Atom | And|Or | Neg | Implies|Equivalent | Forall|Exists | Equal )
         )
    )
```

Entails

```
    in all sublanguages: ( oid?, (if | Rulebase), (then | Rulebase) )
```

Equivalent

```
    attributes: @closure ( + @mapDirection, @mapClosure and @mapMaterial in folog & up)

    in datalog & down and up to dishornlog: ( oid?, ( ( torso, torso) | ( Atom, Atom) ) )

    in hornlogeq: ( oid?, ( (torso, torso) | ( (Atom | Equal), (Atom | Equal) ) ) )

    in hohornlog: ( oid?, ( ( torso, torso) | ( Uniterm, Uniterm ) ) )

    in hohornlogeq: ( oid?, ((torso, torso) | ((Uniterm | Equal), (Uniterm | Equal))) )

    in framehohornlogeq:
    (
       oid?, (
               ( torso, torso) |
               (
                    (Atom | Uniterm | InstanceOf | SubclassOf | Signature | Equal),
                    (Atom | Uniterm | InstanceOf | SubclassOf | Signature | Equal)
               )
             )
    )

    in folog and naffolog:
    (
       oid?, (torso, torso) |
                  (
                      (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists ),
                      (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )
                   )
    )

    in fologeq & naffologeq:
    (
```

```
   oid?, (torso, torso) |
                (
                    (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal),
                    (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal)
                )
)
```

then

```
in datalog & down, nafdatalog, hornlog, and nafhornlog: (Atom)

in negdatalog & nafnegdatalog: (Atom | Neg)

in hornlogeq: (Atom | Equal)

in hohornlog: (Uniterm | Neg )

in hohornlogeq: (Uniterm | Neg | Equal)

in framehohornlogeq: (Atom | Uniterm | InstanceOf | SubclassOf | Signature | Neg | Equal )

in dishornlog: (Atom | Or)

in folog & naffolog: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

in fologeq: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal )

in naffologeq: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal )
```

torso

```
in datalog & down and up to dishornlog: (Atom)

in hornlogeq: ( Atom | Equal )

in hohornlog: ( Uniterm )

in hohornlogeq: ( Uniterm | Equal )

in framehohornlogeq: (Atom | Uniterm | InstanceOf | SubclassOf | Signature | Equal)

in folog and naffolog: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

in fologeq & naffologeq: ( Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal )
```

Rulebase

```
attributes: @closure ( + @mapDirection, @mapClosure and @mapMaterial in folog & up)

in datalog & down and up to dishornlog:
( oid?, ( formula | Atom | Implies | Equivalent | Forall )* )

in hornlogeq:
( oid?, ( formula | Atom | Implies | Equivalent | Forall | Equal )* )

in hohornlog:
( oid?, ( formula | Uniterm | Neg | Implies | Equivalent | Forall )* )

in hohornlogeq:
( oid?, ( formula | Uniterm | Neg | Implies | Equivalent | Forall | Equal )* )


in framehohornlogeq:
( oid?, ( formula|Atom|Uniterm|Neg|Implies|Equivalent|Forall|InstanceOf|SubclassOf|Signature|
Equal )* )

in folog and naffolog:
```

(oid?, (formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists)*)

in fologeq & naffologeq:
(oid?, (formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal)*)

And

attributes within Query only: @closure (+ @mapDirection, @mapClosure and @mapMaterial in folog & up)

in datalog & down, hornlog and dishornlog: (oid?, (formula | Atom | And | Or)*)

in negdatalog: (oid?, (formula | Atom | And | Or | Neg)*)

in nafdatalog: (oid?, (formula | Atom | And | Or | Naf)*)

in nafnegdatalog: (oid?, (formula | Atom | And | Or | Naf | Neg)*)

in hornlogeq: (oid?, (formula | Atom | And | Or | Equal)*)

in nafhornlog: (oid?, (formula | Atom | And | Or | Naf)*)

in hohornlog: (oid?, (formula | Uniterm | And | Or | Neg | Naf)*)

in hohornlogeq: (oid?, (formula | Uniterm | And | Or | Neg | Equal)*)

in framehohornlogeq:
(oid?, (formula|Atom|Uniterm|InstanceOf|SubclassOf|Signature|And|Or|Neg|Naf|Equal)*)

in folog:
(oid?, (formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists)*)

in naffolog:
(oid?, (formula|Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists)*)

in fologeq:
(oid?,(formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal)*)

in naffologeq:
(oid?,(formula|Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists|Equal)*)

Or

attributes within Query only: @closure (+ @mapDirection, @mapClosure and @mapMaterial in folog & up)

in datalog & down, hornlog and dishornlog: (oid?, (formula | Atom | And | Or)*)

in negdatalog: (oid?, (formula | Atom | And | Or | Neg)*)

in nafdatalog: (oid?, (formula | Atom | And | Or | Naf)*)

in nafnegdatalog: (oid?, (formula | Atom | And | Or | Naf | Neg)*)

in hornlogeq: (oid?, (formula | Atom | And | Or | Equal)*)

in nafhornlog: (oid?, (formula | Atom | And | Or | Naf)*)

in hohornlog: ((oid?, (formula | Uniterm | And | Or | Neg | Naf)*)

in hohornlogeq: (oid?, (formula | Uniterm | And | Or | Neg | Equal)*)

in framehohornlogeq:
(oid?, (formula|Atom|Uniterm|InstanceOf|SubclassOf|Signature|And|Or|Neg|Equal)*)

```
in folog:
(oid?,( formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists )* )

in naffolog:
(oid?,( formula|Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists )*)

in fologeq:
(oid?,( formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal )*)

in naffologeq:
(oid?,( formula|Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists|Equal )*)
```

formula

```
(context sensitive)

within And/Or...

   in datalog & down, hornlog and dishornlog: ( Atom | And | Or )

   in negdatalog: ( Atom | And | Or | Neg )

   in nafdatalog: ( Atom | And | Or | Naf )

   in nafnegdatalog: ( Atom | And | Or | Naf | Neg )

   in hornlogeq: ( Atom | And | Or | Equal )

   in nafhornlog: ( Atom | And | Or | Naf )

   in hohornlog: ( Uniterm | And | Or | Neg )

   in hohornlogeq: ( Uniterm | And | Or | Neg | Equal )

   in framehohornlogeq: ( Atom|Uniterm|InstanceOf|SubclassOf|Signature|And|Or|Neg|Equal )

   in folog: ( Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists )

   in naffolog: ( Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists )

   in fologeq: ( Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal )

   in naffologeq: ( Atom|And|Or|Neg|Naf|Implies|Equivalent|Forall|Exists|Equal )
```

@mapMaterial

```
[optional] ( default:yes | no )
```

@material

```
[optional] ( default:yes | no )
```

@mapDirection

```
[optional] ( forward | backward | default:bidirectional )
```

@direction

```
[optional] ( forward | backward | default:bidirectional )
```

@mapClosure

```
[optional] ( universal | existential )
```

@closure

```
[optional] ( universal | existential )
```

Expr

Expr

```
attributes: @type

in hornlog & up (except hohornlog, etc):
(
 oid?, (op | Fun), (slot)*, (resl)?,
 (
  ( ( (arg | Ind | Data | Skolem | Var | Reify | Expr | Plex )+, ( repo )? ) | ( repo) ),
  ( slot )*, ( resl )?
 )?
)
```

op

```
(context sensitive; see also the Atom, Holog and Equality modules)

 within Expr: (Fun)
```

Fun

```
attributes: @iri

in all sublanguages: ( #PCDATA )
```

Plex

```
(context sensitive)

within Atom, Plex, slot...

        in hornlog & up (except hohornlog, etc):
        (
                oid?, (slot)*,
                (
                       ( (arg|Ind|Data|Skolem|Var|Reify|Cterm|Plex)+, (repo)?, (slot)*, (resl)?
                       )? |( (repo), (slot)*, (resl)? )|(resl)
               )
        )

        in hohornlog & up:
        ( (slot)*, (arg | Const | Skolem | Var | Reify | Uniterm)*, (repo)?, (slot)*,(resl)? )

within repo...

        in hornlog & up (except hohornlog, etc):
        ( ( arg | Ind | Data | Skolem | Var | Reify | Expr | Plex | repo )* )

        in hohornlog & up: ( ( arg | Const | Skolem | Var | Reify | Uniterm | repo )* )

within resl...

        in hornlog & up: ( (slot | resl )* )
```

@per

[optional] (default: copy | open | value | effect | model)

Desc

oid

in datalog & down, negdatalog, nafdatalog and nafnegdatalog: (Ind | Data | Var | Skolem | Reify)

in hornlog & up (except hohornlog, etc): (Ind | Data | Var | Skolem | Reify | Expr | Plex)

in hohornlog & up: (Const | Data | Skolem | Var | Reify | Uniterm)

Equality

Equal

```
 in hornlogeq
(
        (oid)?, (degree)?
        (left, right) |
        (  (Ind | Data | Skolem | Var | Reify | Expr | Plex ),
           (Ind | Data | Skolem | Var | Reify | Expr | Plex )
        )
)


in fologeq and naffologeq


(
        (oid)?, (degree)?
        (left, right) |
        (       (Ind | Data | Skolem | Var | Reify | Expr | Plex ),
                (Ind | Data | Skolem | Var | Reify | Expr | Plex )
        )
)




in hohornlogeq
(
        (oid)?, (degree)?
        (left, right) |
        (       ( Const | Skolem | Var | Reify | Uniterm ),
                ( Const | Skolem | Var | Reify | Uniterm )
        )
)


in framehohornlogeq
(
        (oid)?, (degree)?
        (left, right) |
        (       ( Const | Skolem | Var | Reify | Uniterm | Get ),
                ( Const | Skolem | Var | Reify | Uniterm | Get )
        )
)
```

left

```
in hornlogeq
( Ind | Data | Skolem | Var | Reify | Expr | Plex )

in fologeq and naffologeq
( Ind | Data | Skolem | Var | Reify | Expr | Plex )

in hohornlogeq
( Const | Skolem | Var | Reify | Uniterm )

in framehohornlogeq: ( Const | Skolem | Var | Reify | Uniterm | Get )
```

right

```
in hornlogeq
( Ind | Data | Skolem | Var | Reify | Expr | Plex )

in fologeq and naffologeq
( Ind | Data | Skolem | Var | Reify | Expr | Plex )

in hohornlogeq
( Const | Skolem | Var | Reify | Uniterm )

in framehohornlogeq: ( Const | Skolem | Var | Reify | Uniterm | Get )
```

@oriented

```
 [optional] ( default: no | yes )
```

@val

```
[optional] ( default: 0 | 1 )
```

Frame

Set

```
in framehohornlogeq: ( (Const | Skolem | Var | Reify | Uniterm | Get | Set)* )
```

InstanceOf

```
in framehohornlogeq:
( ( Const|Skolem|Var|Reify|Uniterm|Get|Set ),( Const|Skolem|Var|Reify|Uniterm|Get|Set ) )
```

SubclassOf

```
in framehohornlogeq:
( ( Const|Skolem|Var|Reify|Uniterm|Get|Set ),( Const|Skolem|Var|Reify|Uniterm|Get|Set ) )
```

Signature

```
in framehohornlogeq: ( oid, (op | Const | Skolem | Var | Reify | Uniterm)?,slot*)
```

Get

```
in framehohornlogeq: ( oid, SlotProd )
```

SlotProd

```
in framehohornlogeq: ( ( Const | Skolem | Var | Reify | Uniterm | Get | Set)+ )
```

Holog

Uniterm

```
in hohornlog & hohornlogeq:
        (
                (oid)?, ( op|Const|Data|Skolem|Var|Reify|Uniterm ),(slot)*, (resl)?,
                (
                        ( ((arg|Const|Data|Skolem|Var|Reify|Uniterm)+,(repo)?)|(repo)),
                        (slot)*,(resl)?
                )?
        )
 in framehohornlogeq:
        (
           (oid)?, (op|Const|Data|Skolem|Var|Reify|Uniterm),(slot)*, (resl)?,
          (
                ( (( arg|Const|Data|Skolem|Var|Reify|Uniterm|Get )+,(repo)?)|(repo)),
                (slot)*,(resl)?
          )?
        )
```

Atom

```
(context sensitive; see also the Atom module)

within SWSL sublanguages...
in framehohornlogeq: ( oid, ( op | Const | Skolem | Var | Reify | Uniterm )?, slot* )
```

slot

```
(context-sensitive; see also the slot module)

 in framehohornlogeq: ( ( Const | Uniterm ), ( Const | Uniterm | Skolem | Var | Reify )? )
```

op

```
(context sensitive; see also the Atom and Expr modules)

within Uniterm...
in hohornlog & up: ( Const | Skolem | Var | Reify | Uniterm )
```

Const

```
attributes: @iri, @type

in hohornlog & up: ( #PCDATA )
```

@minCard

```
attributes: @minCard

in hohornlog & up: ( #PCDATA )
```

@maxCard

attributes: @maxCard

in hohornlog & up: (#PCDATA)

Naf

Naf

attributes: none (+ @mapDirection and @mapClosure in naffolog & up)

in nafdatalog: (oid?, (weak | Atom))

in nafnegdatalog: (oid?, (weak | Atom | Neg))

in hohornlog (oid?, (weak | Uniterm))

in naffolog: (oid?, (weak|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists))

in naffologeq:(oid?,(weak|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal))

weak

in nafdatalog: (Atom)

in nafnegdatalog: (Atom | Neg)

in hohornlog (Uniterm)

in naffolog: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists)

in naffologeq:(Atom | And| Or| Neg| Implies| Equivalent| Forall| Exists| Equal)

Neg

Neg

attributes: none (+ @mapDirection and @mapClosure in folog & up)

in negdatalog and nafnegdatalog: (oid?, (strong | Atom))

in hohornlog: (oid?, (strong | Uniterm))

in hohornlogeq & up: (oid?, (strong | Uniterm | Equal))

in folog and naffolog: (oid?, (strong | Atom | And|Or | Neg|Implies |Equivalent| Forall|Exists))

in fologeq and naffologeq:
(oid?, (strong| Atom| And|Or |Neg| Implies| Equivalent| Forall| Exists| Equal))

strong

in negdatalog and nafnegdatalog: (Atom)

in hohornlog: (Uniterm)

in hohornlogeq & up: (Uniterm | Equal)

in folog and naffolog: (Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists)

in fologeq and naffologeq: (Atom| And|Or| Neg| Implies| Equivalent | Forall| Exists| Equal)

Performative

RuleML

in all sublanguages: (oid?, (Assert | Query | Protect)*)

Assert

attributes: @mapDirection, @mapClosure and @mapMaterial

in datalog & bindatalog and up to folog: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails | Forall)*)

in bindatagroundlog: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails)*)

in bindatagroundfact: (oid?, (formula | Rulebase | Atom | Entails)*)

in hornlogeq: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails | Forall | Equal)*)

in hohornlog: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall)*)

in hohornlogeq: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall | Equal)*)

in framehohornlogeq:
(oid?,
(formula|Rulebase|Uniterm|Atom|Neg|Implies|Equivalent|Entails|Forall|Equal|InstanceOf|SubclassOf|Signature)*)

in folog and naffolog:
(oid?,(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists)*)

in fologeq and naffologeq:
(oid?,(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|Equals)*)

Retract

attributes: @mapDirection, @mapClosure and @mapMaterial

in datalog & bindatalog and up to folog: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails | Forall)*)

in bindatagroundlog: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails)*)

in bindatagroundfact: (oid?, (formula | Rulebase | Atom | Entails)*)

in hornlogeq: (oid?, (formula | Rulebase | Atom | Implies | Equivalent | Entails | Forall | Equal)*)

in hohornlog: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall)*)

in hohornlogeq: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall | Equal)*)

in framehohornlogeq:
(oid?,
(formula|Rulebase|Uniterm|Atom|Neg|Implies|Equivalent|Entails|Forall|Equal|InstanceOf|SubclassOf|Signature)*)

in folog and naffolog:
(oid?,(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists)*)

in fologeq and naffologeq:
(oid?,(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|Equals)*)

Query

attributes: @closure (+ @mapDirection and @mapClosure in folog & up)

in datalog, bindatalog, hornlog and dishornlog: (oid?, (formula | Rulebase | Atom | And | Or | Entails | Exists)*)

in bindatagroundlog and bindatagroundfact: (oid?, (formula | Rulebase | And | Or | Atom | Entails)*)

in negdatalog: (oid?, (formula | Rulebase | Neg | Atom | And | Or | Entails | Exists)*)

in nafdatalog: (oid?, (formula | Rulebase | Naf | Atom | And | Or | Entails | Exists)*)

in nafnegdatalog: (oid?, (formula | Rulebase | Neg | Naf | Atom | And | Or | Entails | Exists)*)

in hornlogeq: (oid?, (formula | Atom | Rulebase | And | Or | Entails | Exists | Equal)*)

in nafhornlog: (oid?, (formula | Atom | Rulebase | And | Or | Entails | Exists | Naf)*)

in hohornlog: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall)*)

in hohornlogeq: (oid?, (formula | Rulebase | Uniterm | Neg | Implies | Equivalent | Entails | Forall | Equals)*)

in framehohornlogeq:
(oid?,
(formula|Atom|Uniterm|InstanceOf|SubclassOf|Signature|Rulebase|And|Or|Entails|Exists|Neg|Naf|Equal)*)

in folog:
(oid?, (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists)*)

in fologeq:
(oid?, (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists| Equal)*)

in naffolog:
(oid?,
(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|
Naf)*)

in naffologeq:
(oid?,
(formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|
Naf|Equals)*)

formula

within Assert…

in datalog & bindatalog and up to folog: (Atom | Implies | Equivalent | Forall)

in bindatagroundlog: (Rulebase | Atom | Implies | Equivalent | Entails)

in bindatagroundfact: (Rulebase | Atom | Entails)*)

in hornlogeq: (Atom | Implies | Equivalent | Forall | Equal)

in hohornlog: (Uniterm | Implies | Equivalent | Forall)

in hohornlogeq: (Uniterm | Implies | Equivalent | Forall | Equal)

in framehohornlogeq:

```
    ( Atom| Uniterm| InstanceOf| SubclassOf| Signature| Implies| Equivalent| Forall| Equal )

    in folog and naffolog:
    ( Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

    in fologeq and naffologeq:
    ( Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists | Equal )


within Query...

    in datalog, bindatalog, hornlog and dishornlog:
    (Rulebase | Atom | And | Or | Entails | Exists)

    in bindatagroundlog and bindatagroundfact:
    (Rulebase | Atom | And | Or | Entails)

    in negdatalog: (Rulebase | Neg | Atom | And | Or | Entails | Exists)

    in nafdatalog: (Rulebase | Naf | Atom | And | Or | Entails | Exists)

    in nafnegdatalog: ( oid?, (formula | Rulebase | Neg| Naf | Atom | And | Or | Entails | Exists)* )

    in hornlogeq: ( oid?, (formula | Atom | Rulebase | And | Or | Entails |   Exists | Equal)* )

    in nafhornlog: ( (formula | Atom | Rulebase | And | Or | Entails | Exists | Naf)* )

    in hohornlog: (( oid?, (formula | Uniterm | Rulebase | And | Or | Entails | Exists | Neg | Naf)*
    )

    in hohornlogeq: ( oid?, (formula | Uniterm | Rulebase | And | Or | Entails | Exists | Neg | Naf |
    Equal)* )

    in framehohornlogeq:
    (( oid?,
     ( formula | Atom | Uniterm | InstanceOf | SubclassOf | Signature | Rulebase | And | Or | Entails
     |  Exists | Neg |Naf | Equal )*
    )

    in folog:
    ( oid?,
    (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists)* )

    in naffolog:
    ( oid?,
    (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|Naf)* )

    in fologeq:
    ( oid?,
    (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|Equal)* )

    in naffologeq:
    ( oid?,  (formula|Atom|Rulebase|And|Or|Neg|Implies|Equivalent|Entails|Forall|Exists|Naf|Equal)* )
```

Quantifier

Forall

attributes: none (+ @mapDirection and @mapClosure in folog & up)

in bindatalog, datalog & up to (including) hornlog and dishornlog:
(oid?, (declare | Var)+, (formula | Atom | Implies | Equivalent | Forall))

in hornlogeq:
(oid?, (declare | Var)+, (formula | Atom | Implies | Equivalent | Forall | Equal))

in hohornlog: (oid?, (declare | Var)+, (formula | Uniterm | Implies | Equivalent | Forall))

in hohornlogeq: (oid?, (declare | Var)+, (formula |Uniterm | Implies | Equivalent | Forall | Equal))

in framehohornlogeq:
(oid?,(declare|Var)+,
(formula | Atom | Uniterm | InstanceOf | SubclassOf | Signature | Implies | Equivalent | Forall | Equal))

in folog and naffolog:
(oid?, (declare|Var)+, (formula | Atom | And | Or | Neg | Implies | Equivalent | Forall |Exists))

in fologeq and naffologeq:
(oid?, (declare|Var)+, (formula | Atom | And | Or | Neg | Implies | Equivalent | Forall |Exists | Equals))

Exists

attributes: none (+ @mapDirection and @mapClosure in folog & up)

in bindatalog, datalog & up to (including) hornlog and dishornlog:
(oid?, (declare | Var)+, (formula | Atom | And | Or | Exists))

in hornlogeq: (oid?, (declare | Var)+, (formula | Atom | And | Or | Exists | Equal))

in hohornlog: (oid?, (declare | Var)+, (formula | Uniterm | And | Or | Exists)

in hohornlogeq: (oid?, (declare | Var)+, (formula | Uniterm | And | Or | Exists | Equal)

in framehohornlogeq:
(oid?, (declare|Var)+, (formula|Atom|Uniterm|InstanceOf|SubclassOf|Signature|And|Or|Exists|Equal))

in folog and naffolog:
(oid?, (declare | Var)+, (formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists))

in fologeq and naffologeq:
(oid?, (declare | Var)+, (formula|Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal))

declare

in all sublanguages: (Var)

formula

```
 (context sensitive; see also the Connective module)

 within Forall...

    in bindatalog, datalog & up to (including) hornlog and dishornlog:
    (Atom | Implies | Equivalent | Forall)

    in hornlogeq: ( Atom | Implies | Equivalent | Forall | Equal )

    in hohornlog: (Uniterm | Implies | Equivalent | Forall)

    in hohornlogeq: (Uniterm | Implies | Equivalent | Forall | Equal)

    in framehohornlogeq:
    ( Atom|Uniterm|InstanceOf|SubclassOf|Signature|Implies|Equivalent|Forall|Equal )

    in folog and naffolog: ( Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

    in fologeq and naffologeq: (Atom|And|Or|Neg|Implies|Equivalent|Forall|Exists|Equal)

within Exists...

    in bindatalog, datalog & up to (including) hornlog and dishornlog:
    ( Atom | And | Or | Exists )

    in hornlogeq: ( Atom | And | Or | Exists | Equal )

    in hohornlog: ( Uniterm | And | Or | Exists)

    in hohornlogeq: ( Uniterm | And | Or | Exists | Equal)

    in framehohornlogeq:
    (Atom | Uniterm | InstanceOf | SubclassOf | Signature | And | Or | Exists | Equal)

    in folog and naffolog: ( Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists )

    in fologeq and naffologeq: ( Atom | And | Or | Neg | Implies | Equivalent | Forall | Exists
    |Equal )
```

Rest

repo

```
in hornlog & up: ( Var | Plex )
```

resl

```
in hornlog & up: ( Var | Plex )
```

Slot

slot

```
(context sensitive)

  attributes: @card, @weight ( + @minCard and @maxCard in framehohornlogeq)

within Atom, etc...

  in bindatalog, datalog & up to hornlog:
  ( (Ind | Data), ( Ind | Data | Skolem | Var | Reify ) )

  in bindatagroundlog and bindatagroundfact:
  ( ( Ind | Data | Skolem | Reify ),( Ind | Data | Skolem | Reify ) )

  in hornlog & up (except hohornlog, etc):
  ( ( Ind | Data), (Ind | Data | Skolem | Var | Reify | Expr | Plex ) )

  in hohornlog & hohornlogeq:
  ( ( Const | Uniterm ), ( Const | Uniterm | Skolem | Var | Reify ) )

  in framehohornlogeq:
  ( ( Const | Uniterm | Get ), ( Const | Uniterm | Skolem | Var | Reify | Get | Set ) )

within Atom-frame...

  in framehohornlogeq:
  ( ( Const | Uniterm | Get ), ( Const | Uniterm | Skolem | Var | Reify | Get | Set )? )
```

@card

```
[optional] nonNegativeInt
```

@weight

```
[optional] decimal [0,1]
```

Term

arg

attributes: @index

in bindatalog, datalog & up to hornlog: (Ind | Data | Skolem | Var | Reify)

in bindatagroundlog and bindatagroundfact: (Ind | Data | Skolem | Reify)

in hornlog & up (except hohornlog, etc): (Ind | Data | Skolem | Var | Reify | Expr | Plex)

in hohornlog & hohornlogeq: (Const | Skolem | Var | Reify | Uniterm)

in framehohornlogeq: (Const | Skolem | Var | Reify | Uniterm | Get)

Ind

attributes: @iri, @type

in all sublanguages: (#PCDATA)

Data

in all sublanguages: (#PCDATA) [optionally datatyped with XSD built-ins]

Var

attributes: @type

in all sublanguages: (#PCDATA)

Skolem

attributes: @type

in all sublanguages: (#PCDATA)

Reify

in all sublanguages: (<xs:any>?)

@type

[optional] string

@index

[required] positiveInt

Iri

@Iri

[optional] anyURI