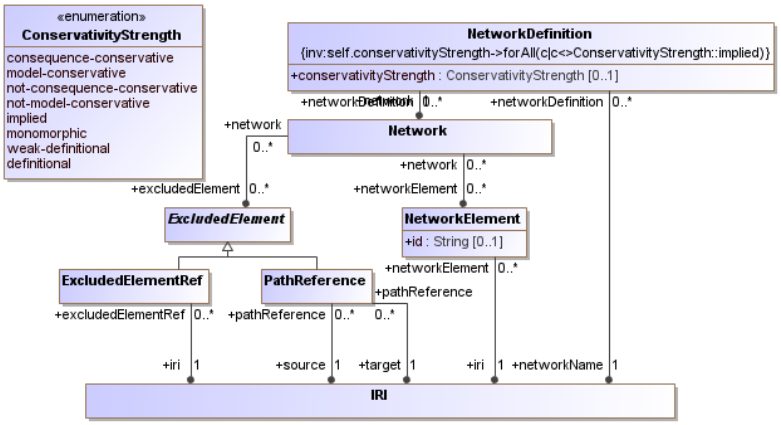

Figure 9.4 – DOL metamodel: Networks

9.4.2 Concrete Syntax

```
NetworkDefinition ::= 'network' NetworkName '='
                      [ConservativityStrength] Network
NetworkName       ::= IRI
Network           ::= NetworkElements [ExcludedElements]
NetworkElements   ::= NetworkElement ( ',' NetworkElement )*
NetworkElement    ::= [Id ':'] ElementRef
ExcludedElements  ::= 'excluding' ExcludedElement ( ',' ExcludedElement )*
ExcludedElement   ::= PathReference | ElementRef
PathReference     ::= IRI '->' IRI
ElementRef        ::= IRI
Id                ::= Letter LetterOrDigit*
```

9.5 OMS

9.5.1 Abstract Syntax

The DOL metamodel for OMS is shown in Fig. 9.5. DOL provides a rich structuring language for OMS, providing extension, translation, unions of OMS and many more. For each of these alternatives, a subclass is introduced. An OMS can be

- a `TranslationOMS` involving both an OMS (to be translated), and a specification of the translation, which is covered by the class `OMSTranslation` (see Appendix M.4, M.8, for examples);
- a `UnionsOMS`, uniting two given OMS (see Appendix M.3 for an example);
- a `ClosureOMS`, applying a closure operator (given by a `Closure`) to an OMS (see Appendix M.5 and M.11 for examples);
- an `ExtensionOMS`, extending a given OMS with another OMS (given by the `Extension`). The major difference between a union and extension is that the members of the unions need to be self-contained OMS, while the extensions may reuse the signature of the extended OMS (see Appendix M.3, M.4, M.5 for examples);
- an `ExtendingOMS`, which is a very simple form of OMS, namely a basic OMS or an OMS reference (see below);
- a `FilteringOMS`, applying a filtering operator (given by a `Filtering`) to an OMS (see Appendix M.9.1.4 for an example);
- an `ApproximationOMS`, applying an approximation operator (given by an `Approximation`) to an OMS (see Appendix M.9 for an example);
- a `CombinationOMS`, giving a combination of (the OMS contained in) an OMS network (technically, this is a colimit, see [**?**]) (see Appendix M.7 for an example of the use of `combination`);

— both `%satisfiable`, `%cons` and `%mcons` are translated to `model-conservative`,

— both `%consistent` and `%ccons` are translated to `consequence-conservative`,

— both `%unsatisfiable` and `%notmcons` are translated to `not-model-conservative`,

— both `%inconsistent` and `%notccons` are translated to `not-consequence-conservative`,

— moreover, both `closed-world` and `minimize` are translated to `minimize`.

Note that the MOF abstract syntax subsumes all these elements except from those in the last line under the enumeration class `ConservativityStrength`. Not all elements of the enumeration can be used at any position; the corresponding restrictions are expressed as OCL constraints. By contrast, the concrete syntax features a more fine-grained structure of non-terminals (`Conservative`, `ConservativityStrength` and `ExtConservativityStrength`) in order to express the same constraints via the EBNF grammar.

```
BasicOMS            ::= <language and serialization specific>
ClosableOMS         ::= BasicOMS | OMSRef [ImportName]
ExtendingOMS        ::= ClosableOMS | RelativeClosureOMS
RelativeClosureOMS  ::= ClosureType '{' ClosableOMS '}'
OMS                 ::= ExtendingOMS
                      | OMS Closure
                      | OMS OMSTranslation
                      | OMS Reduction
                      | OMS Extraction
                      | OMS Approximation
                      | OMS Filtering
                      | OMS 'and' [ConservativityStrength] OMS
                      | OMS 'then' ExtensionOMS
                      | Qualification* ':' GroupOMS
                      | 'combine' NetworkElements [ExcludeExtensions]
                      | GroupOMS
Closure             ::= ClosureType CircMin [CircVars]
ClosureType         ::= 'minimize'
                      | 'closed-world'
                      | 'maximize'
                      | 'free'
                      | 'cofree'
CircMin             ::= Symbol Symbol*
CircVars            ::= 'vars' (Symbol Symbol*)
GroupOMS            ::= '{' OMS '}' | OMSRef
OMSTranslation      ::= 'with' LanguageTranslation* SymbolMap
                      | 'with' LanguageTranslation+
LanguageTranslation ::= 'translation' OMSLanguageTranslation
Reduction           ::= 'hide' LogicReduction* SymbolList
                      | 'hide' LogicReduction+
                      | 'reveal' SymbolList
LogicReduction      ::= 'along' OMSLanguageTranslation
SymbolList          ::= Symbol ( ',' Symbol )*
SymbolMap           ::= GeneralSymbolMapItem ( ',' GeneralSymbolMapItem )*
Extraction          ::= 'extract' InterfaceSignature
                      | 'remove' InterfaceSignature
Approximation       ::= 'forget' InterfaceSignature ['keep' LogicRef]
                      | 'keep' InterfaceSignature ['keep' LogicRef]
                      | 'keep' LogicRef
Filtering           ::= RemovalKind BasicOMSOrSymbolList
RemovalKind         ::= 'reject' | 'select'
BasicOMSOrSymbolList ::= '{' BasicOMS '}' | SymbolList
ExtensionOMS        ::= [ExtConservativityStrength]
                        [ExtensionName]
                        ExtendingOMS
ConservativityStrength ::= Conservative | '%mono' | '%wdef' | '%def'
ExtConservativityStrength ::= ConservativityStrength | '%implied'
Conservative        ::= '%cons'
```

```
                    | '%ccons'
                    | '%mcons'
                    | '%notccons'
                    | '%notmcons'
                    | '%consistent'
                    | '%inconsistent'
                    | '%satifsiable'
                    | '%unsatisfiable'
InterfaceSignature ::= SymbolList
ImportName         ::= '%(' IRI ')%'
ExtensionName      ::= '%(' IRI ')%'
OMSkeyword         ::= 'ontology'
                     | 'onto'
                     | 'specification'
                     | 'spec'
                     | 'model'
                     | 'oms'
OMSDefinition      ::= OMSkeyword OMSName '='
                       [ConservativityStrength] OMS 'end'
Symbol             ::= IRI
SymbolMapItem      ::= Symbol '|->' Symbol
GeneralSymbolMapItem ::= Symbol | SymbolMapItem
Sentence           ::= <an expression specific to an OMS language>
OMSName            ::= IRI
OMSRef             ::= IRI
LoLaRef            ::= LanguageRef | LogicRef


OMSLanguageTranslation ::= OMSLanguageTranslationRef | '->' LoLaRef
OMSLanguageTranslationRef ::= IRI
```

The above grammar allows for some grouping ambiguity when using operators in OMS definitions. These ambiguities are resolved according to the following list, listing operators in decreasing order of precedence:

- `minimize`, `maximize`, `free`, and `cofree`.
- `extract`, `forget`, `hide`, `keep`, `reject`, `remove`, `reveal`, `select`, and `with`.
- `and`.
- `then`.

Multiple occurrences of the same operator are grouped in a left associative manner. In all other cases operators on the same precedence level are not implicitly grouped and have to be grouped explicitly. Omitting such an explicit grouping results in a parse error.

9.6 OMS Mappings

9.6.1 Abstract Syntax

An OMS mapping provides a connection between two OMS. An OMS mapping definition is the definition of either a named interpretation (`InterpretationDefinition`, see Annex M.3 for an example), entailment (`EntailmentDefinition`, see use case 7.11.1 for an example), refinement (`RefinementDefinition`, see use cases 7.10 and 7.11.1 for examples) or equivalence (`EquivalenceDefinition`, see Annex M.3 for an example), a named declaration of the relation between a module of an OMS and the whole OMS (`ConservativeExtensionDefinition`, see use case 7.4 for an example), or a named alignment (`AlignmentDefinition`, see use case 7.3 and Annex M.7 for examples).

The DOL metamodel for interpretations and refinements is shown in Fig. 9.8. Both interpretations and refinements specify a logical entailment or specialization relation between OMS.

```
AlignmentCardinality ::= '1' | '?' | '+' | '*'
AlignmentType        ::= GroupOMS 'to' GroupOMS
AlignmentSemantics   ::= 'SingleDomain'
                       | 'GlobalDomain'
                       | 'ContextualizedDomain'
Correspondence       ::= CorrespondenceBlock | SingleCorrespondence | DefaultCorrespondence
DefaultCorrespondence ::= '*'
CorrespondenceBlock  ::= 'relation' [Relation] [Confidence] '{'
                          Correspondence ( ',' Correspondence )* '}'
SingleCorrespondence ::= Symbol [Relation] [Confidence]
                          GeneralizedTerm [CorrespondenceId]
CorrespondenceId     ::= '%(' IRI ')%'
Symbol             ::= IRI
GeneralizedTerm      ::= Symbol
Relation             ::= RelationReference | StandardRelation
RelationReference    ::= IRI
StandardRelation     ::= '>' | '<' | '=' | '%' | 'ni' | 'in'
                         < No keyword corresponding to default-relation as this is just the default if
                           Relation is omitted >
Confidence           ::= Double
Double               ::= < a number ∈ [0, 1] >
```

9.7 Identifiers

This section specifies the abstract syntax of identifiers of DOL OMS and their elements. Further, it introduces the concrete syntax that is used in the DOL serialization.

9.7.1 IRIs

In accordance with best practices for publishing OMS on the Web, identifiers of OMS and their elements **should** not just serve as *names*, but also as *locators*, which, when dereferenced, give access to a concrete representation of an OMS or one of its elements. (For the specific case of RDF Schema and OWL OMS, these best practices are documented in [**?**]. The latter is a specialization of the linked data principles, which apply to any machine-processable data published on the Web [**?**].) It is recommended that publicly accessible DOL OMS be published as linked data.

Therefore, in order to impose fewer conformance requirements on applications, DOL requires the use of IRIs for identification per **NR11**. It is **recommended** that DOL libraries use IRIs that translate to URLs when applying the algorithm for mapping IRIs to URIs specified in **NR11**, Section 3.1. DOL descriptions of any element of a DOL library that is identified by a certain IRI **should** be *located* at the corresponding URL, so that agents can locate them. As IRIs are specified with a concrete syntax only in **NR11**, DOL adopts the latter into its abstract syntax as well as all of its concrete syntaxes (serializations). The DOL metamodel for IRIs and prefixes is shown in Fig. 9.12.

In accordance with semantic web best practices such as the OWL Manchester Syntax [**?**], this OMG Specification does not allow relative IRIs, and does not offer a mechanism for defining a base IRI, against which relative IRIs could be resolved.

Concerning these languages, note that they allow arbitrary IRIs in principle, but in practice they strongly recommend using IRIs consisting of two components [**?**]:

namespace: an IRI that identifies an OMS, usually ending with `#` or `/`. (See annex O for a specific linked-data compliant URL scheme for DOL.)

local name: a name that identifies a non-logical symbol within an OMS

else
$P \leftarrow$ the innermost prefix map in D, starting from the place of O inside D, and going up the abstract syntax tree towards the root of D
while P is defined **do**
if P binds p to an IRI pi **then**
$nsi \leftarrow pi$
break out of the **while** loop
end if
$P \leftarrow$ the next prefix map in D, starting from the place of the current P inside D, and going up the abstract syntax tree towards the root of D
end while
return an error
end if
$i \leftarrow concat(nsi, ref)$
else
return an error
end if
end if
return i

This mechanism applies to basic OMS given inline in a DOL library (`BasicOMS`), not to OMS in external documents (`NativeDocument`); the latter **shall** be self-contained.

While CURIEs used for identifying parts of a DOL library (cf. clause 9.7.2) are merely syntactic sugar, the prefix map for a basic OMS is essential to determining the semantics of the basic OMS within the DOL library.

9.7.4 Concrete Syntax

```
IRI            ::= '<' FullIRI '>' | CURIE
FullIRI        ::= < an IRI as defined in NR11 >
CURIE          ::= MaybeEmptyCURIE -
MaybeEmptyCURIE ::= [Prefix] RefWithoutComma
RefWithoutComma ::= Reference - StringWithComma
StringWithComma ::= UChar* ',' UChar*
UChar          ::= < any Unicode NR17 character >
Prefix         ::= NCName ':'< see "NCName" in NR15, Section 3 >
Reference      ::= Path [Query] [Fragment]
Path           ::= ipath-absolute | ipath-rootless | ipath-empty < as defined in NR11 >
Query          ::= '?' iquery < as defined in NR11 >
Fragment       ::= '#' ifragment < as defined in NR11 >
```

In a CURIE without a prefix, the `reference` part is **not allowed** to match any of the keywords of the DOL syntax (cf. clause 9.8.1).

9.8 Lexical Symbols

The character set for the DOL text serialization is the UTF-8 encoding of Unicode **NR17**. However, OMS can always be input in the Basic Latin subset, also known as US-ASCII.[17)] For enhanced readability of OMS, the DOL text serialization particularly supports the native Unicode glyphs that represent common mathematical symbols (e.g. Greek letters) or operators (e.g. ∂ for partial derivatives).

[17)] In this case, IRIs will have to be mapped to URIs following section 3.1 of **NR11**.

```
Network             ::= network NetworkElement* ExcludedElement*
NetworkElement      ::= network-element [Id] IRI
ExcludedElement     ::= PathReference | ExcludedElementRef
PathReference       ::= path IRI IRI
ExcludedElementRef  ::= IRI
```

K.4 OMS

```
BasicOMS           ::= < language specific >
OMS                ::= ExtendingOMS
                     | ClosureOMS
                     | TranslationOMS
                     | ReductionOMS
                     | ExtractionOMS
                     | ApproximationOMS
                     | FilteringOMS
                     | UnionOMS
                     | ExtensionOMS
                     | QualifiedOMS
                     | CombinationOMS
                     | ApplicationOMS
ClosureOMS         ::= closure-symbols OMS Closure
TranslationOMS     ::= translation OMS OMSTranslation
ReductionOMS       ::= reduction OMS Reduction
ExtractionOMS      ::= module-extract OMS Extraction
ApproximationOMS   ::= approximation OMS Approximation
FilteringOMS       ::= filtering OMS Filtering
UnionOMS           ::= union OMS [ConservativityStrength] OMS
ExtensionOMS       ::= extension OMS Extension
QualifiedOMS       ::= qualified-oms Qualification Qualification* OMS
CombinationOMS     ::= combination Network
ApplicationOMS     ::= application OMS SubstName
OMSDefinition      ::= oms-definition OMSName [ConservativityStrength] OMS
ConservativityStrength ::= consequence-conservative
                     | model-conservative
                     | not-consequence-conservative
                     | not-model-conservative
                     | implied
                     | monomorphic
                     | weak-definitional
                     | definitional
OMSName            ::= IRI
SubstName          ::= IRI

OMSReference       ::= oms-reference OMSRef [ImportName]
Extension          ::= extension [ConservativityStrength]
                               [ExtensionName] ExtendingOMS
ExtendingOMS       ::= ClosableOMS | RelativeClosureOMS
RelativeClosureOMS ::= relative-closure ClosureType ClosableOMS
Closure            ::= ClosureType CircClosure CircVars
ClosureType        ::= minimize | maximize | free | cofree
CircClosure        ::= Symbol Symbol*
CircVars           ::= Symbol*
ExtensionName      ::= IRI
ImportName         ::= IRI
OMSRef             ::= IRI

Reduction          ::= reduction RemovalKind OMSLanguageTranslation*
                               [SymbolList]
SymbolList         ::= Symbol Symbol*
SymbolMap          ::= symbol-map GeneralSymbolMapItem
```

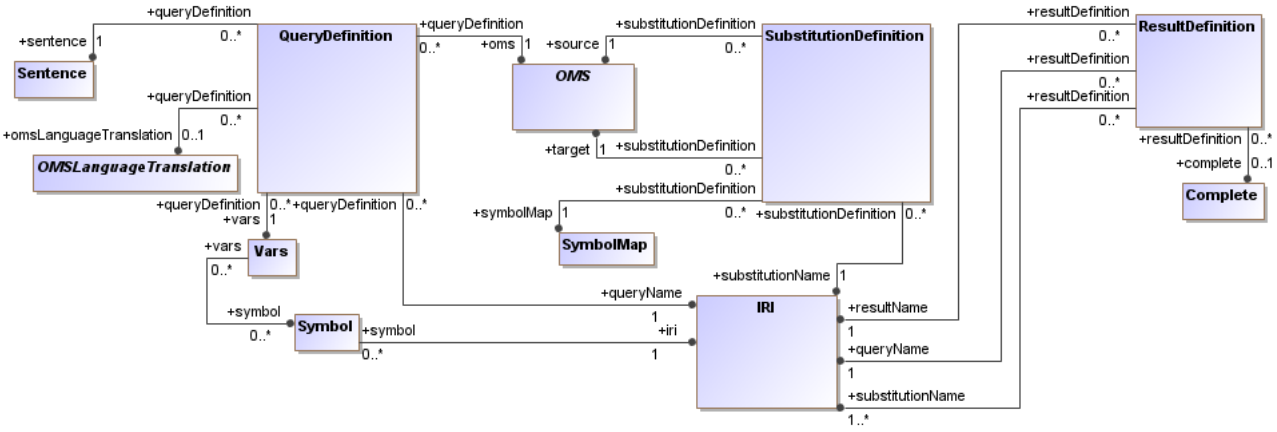

Figure L.1 – Extension of DOL metamodel with queries

```
ResultDefinition    ::= 'result' ResultName '=' SubstitutionName
                        ( ',' SubstitutionName )* 'for' QueryName
                        ['%complete'] 'end'
OMS                 ::= ... | OMS 'with' SubstitutionName
QueryName           ::= IRI
SubstitutionName    ::= IRI
ResultName          ::= IRI
Vars                ::= Symbol ( ',' Symbol )*
```

L.5 EBNF Abstract Syntax

```
QueryRelatedDefinition ::= QueryDefinition
                         | SubstitutionDefinition
                         | ResultDefinition
QueryDefinition    ::= select-query-definition
                                          QueryName Vars Sentence OMS
                                          [OMSLanguageTranslation]
SubstitutionDefinition ::= substitution-definition
                                             SubstitutionName OMS OMS
                                             SymbolMap
ResultDefinition   ::= result-definition ResultName
                                         SubstitutionName SubstitutionName*
                                         QueryName [Complete]
Sentence           ::= < an expression specific to an OMS language >
OMS                ::= ... | application  OMS  SubstitutionName
QueryName          ::= IRI
SubstitutionName   ::= IRI
ResultName         ::= IRI
Vars               ::= Symbol*
Complete           ::= complete
```

L.6 Semantics of Queries

While queries are very important from a practical point of view, their semantics so far has been developed only for individual institutions. In [?], three options for an institution-independent semantics of queries and derived signature morphisms (which can map symbols to terms) are discussed. Currently, it is not clear which one would be the best choice. It is expected that after some experience with DOL, a choice will crystallize. This means that in the current version, the semantics of queries is elided, and left for a later version of DOL.