1 Scope

1.1 General

This OMG Specification specifies the Distributed Ontology, Modeling and Specification Language (DOL). DOL is designed to achieve integration and interoperability of ontologies, specifications and ~~MDE~~ models (OMS for short). DOL is a language for distributed knowledge representation, system specification and model-driven development across multiple OMS, particularly OMS that have been formalized in different OMS languages. This OMG Specification responds to the OntoIOp Request for Proposals [22].

1.2 Background Information

Logical languages are used in several fields of computing for the development of formal, machine-processable texts that carry a formal semantics. Among those fields are 1) **O**ntologies formalizing domain knowledge, 2) (formal) **M**odels of systems, and 3) the formal **S**pecification of systems. Ontologies, ~~MDE~~ models and specifications will (for the purpose of this document) henceforth be abbreviated as **OMS**.

An OMS provides formal descriptions, which range in scope from domain knowledge and activities (ontologies, ~~MDE~~ models) to properties and behaviors of hardware and software systems (~~MDE~~ models, specifications). These formal descriptions can be used for the analysis and verification of domain models, system models and systems themselves, using rigorous and effective reasoning tools. As systems increase in complexity, it becomes concomitantly less practical to provide a monolithic logical cover for all. Instead various ~~MDE~~ models are developed to represent different viewpoints or perspectives on a domain or system. Hence, interoperability becomes a crucial issue, in particular, formal interoperability, i.e. interoperability that is based on the formal semantics of the different viewpoints. Interoperability is both about the ability to interface different domains and systems and the ability to use several OMS in a common application scenario. Further, interoperability is about coherence and consistency, ensuring at an early stage of the development that a coherent system can be reached.

In complex applications, which involve multiple OMS with overlapping concept spaces, it is often necessary to identify correspondences between concepts in the different OMS; this is called OMS alignment. While OMS alignment is most commonly studied for OMS formalized in the same OMS language, the different OMS used by complex applications may also be written in different OMS languages, which may even vary in their expressiveness. This OMG Specification faces this diversity not by proposing yet another OMS language that would subsume all the others. Instead, it accepts the diverse reality and formulates means (on a sound and formal semantic basis) to compare and integrate OMS that are written in different formalisms. It specifies DOL, a formal language for expressing not only OMS but also mappings between OMS formalized in different OMS languages.

Thus, DOL gives interoperability a formal grounding and makes heterogeneous OMS and services based on them amenable to checking of coherence (e.g. consistency, conservativity, intended consequences, and compliance).

1.3 Features Within Scope

The following are within the scope of this OMG Specification:

1) homogeneous OMS as well as heterogeneous OMS (OMS that consist of parts written in different languages);
2) mappings between OMS (which map OMS symbols to OMS symbols);
3) OMS networks (involving several OMS and mappings between them);
4) translations between different OMS languages conforming with DOL (translating a whole OMS to another language);
5) structuring constructs for modeling non-monotonic behavior;
6) annotation and documentation of OMS, mappings between OMS, symbols, and sentences;
7) recommendations of vocabularies for annotating and documenting OMS;
8) a syntax for embedding the constructs mentioned under (1)–(6) as annotations into existing OMS;
9) a syntax for expressing (1)–(5) as standoff markup that points into existing OMS;

4 Terms and Definitions

4.1 Distributed Ontology, Modeling and Specification Language

Distributed Ontology, Modeling and Specification Language
DOL unified metalanguage for the structured and heterogeneous expression of ontologies, specifications, and models, using DOL libraries of OMS, OMS mappings and OMS networks, whose syntax and semantics are specified in this OMG Specification.

DOL library collection of named OMS and OMS networks, possibly written in different OMS languages, linked by named OMS mappings.

4.2 Native OMS, OMS, and OMS Languages

native OMS collection of expressions (like non-logical symbols, sentences and structuring elements) from a given OMS language.
Example A UML class model, an ontology written in OWL 2 EL, and a specification written in CASL are three different native OMS.
Note An OMS can be written in different OMS language serializations.

native document document containing a native OMS.

DOL document document containing a DOL library.

OMS language language equipped with a formal, declarative, logic-based semantics, plus non-logical annotations.
Example OMS languages include OWL 2 DL, Common Logic, F-logic, UML class models, RDF Schema, and OBO.
Note An OMS language is used for the formal specification of native OMS.
Note An OMS language has a logical language aspect, a structuring language aspect, and an annotation language aspect.

DOL structured OMS syntactically valid DOL expression denoting an OMS that is built from smaller OMS as building blocks.

Note DOL structured OMS, typically, use basic OMS as building blocks for defining other structured OMS, OMS mappings or OMS networks.
Note All DOL structured OMS are structured OMS.

ontology logical theory that is used as a shard conceptualization

model logical theory that is used as an abstract representation of a domain or of a system, in the sense of model-driven engineering (MDE)
Note Not to be confused with the term model in the sense of logic (model theory). In this document, we use the term realization for models in the sense of model theory in logic.

specification logical theory that is used to express formal constraint in mathematical structures, software systems and/or hardware systems

~~**OMS (ontology, specification or MDE model)**~~ **OMS (ontology, specification or model)** basic OMS or structured OMS.
Note

An OMS is either a basic OMS (which is always a native OMS, and can occur as a text fragment in a DOL document) or a structured OMS (which can be either a native structured OMS contained in some native document, or a DOL structured OMS contained in a DOL document).

Note An OMS has a single signature and model class over that signature as its model-theoretic semantics.

basic OMS
flat OMS native OMS that does not utilize any elements from the structuring language aspect of its language.
Note Basic OMS are self-contained in the sense that their semantics does not depend on some other OMS. In particular, a basic OMS does not involve any imports.
Note Since a basic OMS has no structuring elements, it consists of (or at least denotes) a signature equipped with a set of sentences and annotations.
Note In signature-free logics like Common Logic or TPTP, a basic OMS only consists of sentences. A signature can be obtained a posteriori by collecting all non-logical symbols occuring in the sentences.

~~**non-logical symbol**OMS symbol atomic expression or syntactic constituent of an OMS that requires an interpretation through a mo~~
non-logical symbolOMS symbol atomic expression or syntactic constituent of an OMS that requires an interpretation through a rea

Note This differs from the notion of “atomic sentence”: such sentences may involve several non-logical symbols.

Example Non-logical symbols in OWL **NR2** (there called “entities”) comprise

— individuals (denoting objects from the domain of discourse),
— classes (denoting sets of objects; also called concepts), and
— properties (denoting binary relations over objects; also called roles).

These non-logical symbols are distinguished from logical symbols in OWL, e.g., those for intersection and union of classes.

Example Non-logical symbols in Common Logic **NR7** comprise

— names (denoting objects from the domain of discourse),
— sequence markers (denoting sequences of objects).

These non-logical symbols are distinguished from logical symbols in Common Logic, e.g. logical connectives and quantifiers.

signature
vocabulary set (or otherwise structured collection) of non-logical symbols of an OMS.
Note The signature of a term is the set of all non-logical symbols occurring in the term. The notion of signature depends on the OMS language or logic.
Note The signature of an OMS is usually unequivocally determinable.

realization semantic interpretation of all non-logical symbols of a signature.
Note A ~~model~~ realization of an OMS is a ~~model~~ realization of the signature of the OMS that also satisfies all the additional constraints expressed by the OMS. In case of flattenable OMS, these constraints are expressed by the axioms of the OMS.
Note ~~This term refers to *model* in the sense of model theory (a branch of logic). It is not to be confused with in the sense of modeling (i.e. , the “M” in OMS). The notion of model depends on the~~ The notion of realization depends on the OMS language or logic.
Note In logical model theory, a realization is called “model”. However, we have reserved the term “model” for models in the sense of model-driven engineering.

expression a finite combination of symbols that are well-formed according to applicable rules (depending on the language)

term syntactic expression either consisting of a single non-logical symbol or recursively composed of other terms (a.k.a. its subterms).
Note A term belongs to the logical language aspect of an OMS language.

sentence term that is either true or false in a given realization, i.e. which is assigned a truth value in this realization.
Note In a realization, on the one hand, a sentence is always true or false. In an OMS, on the other hand, a sentence can

have several logical statuses. For example, a sentence can be: an axiom, if postulated to be true; a theorem, if proven from other axioms and theorems; or a conjecture, if expecting to be proven from other axioms and theorems.
NOTE A sentence can conform to one or more signatures (namely those signatures containing all non-logical symbols used in the sentence).
NOTE It is quite common that sentences are required to be closed (i.e. have no free variables). However, this depends on the OMS language at hand.
NOTE A sentence belongs to the logical language aspect of an OMS language.
NOTE The notion of sentence depends on the OMS language or logic.

satisfaction relation relation between realizations and sentences indicating which sentences hold true in the realization.
NOTE The satisfaction relation depends on the OMS language or logic.

logical theory signature equipped with a set of sentences over the signature.

NOTE Each logical theory can also be written a basic OMS, and conversely each basic OMS has as its semantics a logical theory.

entailment
logical consequence
specialization relation between two OMS (or an OMS and a sentence, or two OMS networks, or an OMS network and an OMS) expressing that the second item (the conclusion) is logically implied by the first one (the premise).
NOTE Entailment expresses that each ~~model~~ realization satisfying the premise also satisfies the conclusion.

NOTE The converse is generalization.

axiom sentence that is postulated to be valid (i.e. true in every realization).

theorem sentence that has been proven from other axioms and theorems and therefore has been demonstrated to be a logical consequence of the axioms.

tool software for processing DOL libraries and OMS. **theorem proving** process of demonstraing that a sentence (or OMS) is the logical consequence of some OMS.

theorem prover tool implementing theorem proving.

4.3 Structured OMS

structured OMS OMS that results from other (basic and structured) OMS by import, union, combination, OMS translation, OMS reduction or other structuring operations.
NOTE Structured OMS are either DOL structured OMS or native OMS that utilize elements of the structuring language aspect of their OMS language.

flattenable OMS OMS that can be seen, by purely syntactical means, to be logically equivalent to a flat OMS.
NOTE More precisely, an OMS is flattenable if and only if it is either a basic OMS or it is an extension, union, translation, module, approximation, filtering, or reference of named OMS involving only flattenable OMS.

elusive OMS OMS that is not flattenable.

subOMS OMS whose associated sets of non-logical symbols and sentences are subsets of those present in a given larger OMS.

import reference to an OMS behaving as if it were verbatim included; also import of DOL libraries.
NOTE Semantically, an import of O_2 into O_1 is equivalent to the verbatim inclusion of O_2 in place of the import declaration.
NOTE The purpose of O_2 importing O_1 is to make non-logical symbols and sentences of O_1 available in O_2.
NOTE Importing O_1 into O_2 turns O_2 into an extension of O_1.

NOTE An owl:import in OWL is an import.
NOTE The import of a whole DOL library into another DOL library is also called import.

union DOL structured OMS expressing the aggregation of several OMS to a new OMS, without any renaming.

OMS translation DOL structured OMS expressing the assignment of new names to some non-logical symbols of an OMS, or translation of an OMS along a language translation.
NOTE An OMS translation results in an OMS mapping between the original and the renamed OMS.
NOTE Typically, the resulting OMS mapping of a translation is surjective: the symbols of the original OMS can be identified by the renaming, but no new symbols are added.

OMS reduction DOL structured OMS expressing the restriction of an OMS to a smaller signature.

local environment context for an OMS, being the signature built from all previously-declared symbols and axioms.

extension structured OMS extending a given OMS with new symbols and sentences.
NOTE The new symbols and sentences are interpreted relative to the local envorinment, which is the signature of the "given OMS".

extension mapping inclusion OMS mapping between two OMS where the sets of non-logical symbols and sentences of the second OMS are supersets of those present in the first OMS.
NOTE The second OMS is said to extend the first, and is an extension of the first OMS.

conservative extension extension that does not add new logical properties with respect to the signature of the extended OMS.
NOTE An extension is a consequence-theoretic or model-theoretic conservative extension. If used without qualification, the model-theoretic version is meant.

consequence-theoretic conservative extension extension that does not add new theorems (in terms of the unextended signature).
NOTE An extension O_2 of an OMS O_1 is a consequence-theoretic conservative extension, if all properties formulated in the signature of O_1 hold for O_1 whenever they hold for O_2.

~~**model-theoretic conservative extension** extension that does not lead to a restriction of class of models of an OMS.~~
model-theoretic conservative extension extension that does not lead to a restriction of class of realizations of an OMS.

NOTE An extension O_2 of an OMS O_1 is a model-theoretic conservative extension, if each ~~model~~ realization of O_1 can be expanded to a ~~model~~ realization of O_2.
NOTE Each model-theoretic conservative extension is also a consequence-theoretic one, but not vice versa.

monomorphic extension extension whose newly introduced non-logical symbols are interpreted in a way unique up to isomorphism.
NOTE An extension O_2 of an OMS O_1 is a monomorphic extension, if each ~~model~~ realization of O_1 can be expanded to a ~~model~~ realization of O_2 that is unique up to isomorphism.
NOTE Each monomorphic extension is also a model-theoretic conservative extension but not vice versa.

definitional extension extension whose newly introduced non-logical symbols are interpreted in a unique way.
NOTE An extension O_2 of an OMS O_1 is a definitional extension, if each ~~model~~ realization of O_1 can be uniquely expanded to a ~~model~~ realization of O_2.
NOTE O_2 being a definitional extension of O_1 implies a bijective correspondence between the classes of ~~models~~ realizations of O_2 and O_1.
NOTE Each definitional extension is also a monomorphic extension but not vice versa.

weak definitional extension extension whose newly introduced non-logical symbols can be interpreted in at most one way.
NOTE An extension O_2 of an OMS O_1 is a weak definitional extension, if each ~~model~~ realization of O_1 can be expanded to at most one ~~model~~ realization of O_2.
NOTE An extension is definitional if and only if it is both weakly definitional and model-theoretically conservative.

implied extension model-theoretic conservative extension that does not introduce new non-logical symbols.

NOTE A conservative extension O_2 of an OMS O_1 is an implied extension, if and only if the signature of O_2 is the signature of O_1. O_2 is an implied extension of O_1 if and only if the ~~model class of~~ class of realizations of O_2 is the ~~model class of~~ class of realizations of O_1.
NOTE Each implied extension is also a definitional extension but not vice versa.

consistency property of an OMS expressing that it has a non-trivial set of logical consequences in the sense that not every sentence follows from the OMS.
NOTE The opposite is inconsistency.
NOTE In many (but not all) logics, consistency of an OMS equivalently can be defined as *false* not being a logical consequence of the OMS. However, this does not work for logics that e.g. do not feature a *false*. See [59] for a more detailed discussion.

satisfiability property of an OMS expressing that it is satisfied by least one realization.
NOTE The opposite is unsatisfiability.
NOTE Any satisfiable OMS is consistent, but there are some logics where the converse does not hold.

model finding process that finds realizations (models) of an OMS and thus proves it to be satisfiable.

model finder tool that implements model finding.

module structured OMS expressing a subOMS that conservatively extends to the whole OMS.
NOTE The conservative extension can be either model-theoretic or consequence-theoretic; without qualification, the model-theoretic version is used.

module extraction activity of obtaining from an OMS concrete modules to be used for a particular purpose (e.g. to contain a particular sub-signature of the original OMS).
NOTE Cited and slightly adapted from [71].
NOTE The goal of module extraction is “decomposing an OMS into smaller, more manageable modules with appropriate dependencies” [70].
EXAMPLE Assume one extracts a module about white wines from an OWL DL ontology about wines of any kind. That module would contain the declaration of the non-logical symbol “white wine”, all declarations of non-logical symbols related to “white wine”, and all sentences about all of these non-logical symbols.

approximant logically implied theory (possibly after suitable translation) of an OMS in a smaller signature or a sublanguage.

maximum approximant best possible approximant of an OMS in a smaller signature or a sublanguage.
NOTE Technically, a maximum approximant is a uniform interpolant, see [46].

approximation structured OMS that expresses a maximum approximant.

filtering structured OMS expressing the verbatim removal of symbols or axioms from an OMS.
NOTE If a symbol is removed, all axioms containing that symbol are removed, too.

closed world assumption assumption that facts whose status is unknown are true.

closure
circumscription structured OMS expressing a variant of the closed world assumption by restricting the realizations to those that are minimal, maximal, free or cofree (with respect to the local environment).
NOTE Symbols from the local environment are assumed to have a fixed interpretation. Only the symbols newly declared in the closure are forced to have minimal or maximal interpretation.
NOTE DOL supports four different forms of closure: minimization and maximization as well as freeness and cofreeness (explained below).

NOTE See [50], [43].

minimization form of closure that restricts the ~~models~~ realizations to those that are minimal (with respect to the local environment). **maximization** form of closure that restricts the ~~models~~ realizations to those that are maximal (with respect to the local environment).

4.7 Interoperability

logically interoperable property of structured OMS, which may be written in different OMS languages supporting different logics, of being usable jointly in a coherent way (via suitable OMS language translations), such that the notions of their overall consistency and logical entailment have a precise logical semantics.
NOTE Within ISO 19763 and ISO 20943, metamodel interoperability is equivalent to the existence of mapping, which are statements that the domains represented by two ~~MDE~~ models intersect and there is a need to register details of the correspondence between the structures in the ~~MDE~~ models that semantically represent this overlap. Within these standards, an ~~MDE~~ model is a representation of some aspect of a domain of interest using a normative modeling facility and modeling constructs.

The notion of logical interoperability is distinct from the notion of interoperability used in ISO/IEC 2381-1 Information Technology Vocabulary – Part 1: Fundamental Terms, which is restricted to the capability to communicate, execute programs, or transfer data among various hardware or software entities in a manner that requires the user to have little or no knowledge of the unique characteristics of those entities.

OMS interoperability relation among OMS (via OMS alignments) which are logically interoperable.

4.8 Abstract and Concrete Syntax

concrete syntax
serialization specific syntactic encoding of a given OMS language or of DOL.
NOTE Serializations serve as standard formats for exchanging DOL documents and OMS between human beings and tools.
EXAMPLE OWL uses the term "serialization"; the following are standard OWL serializations: OWL functional-style syntax, OWL/XML, OWL Manchester syntax, plus any standard serialization of RDF (e.g. RDF/XML, Turtle, . . .). However, W3C specifications only require an RDF/XML implementation for OWL2 tools.
EXAMPLE Common Logic uses the term "dialect"; the following are standard Common Logic dialects: Common Logic Interchange Format (CLIF), Conceptual Graph Interchange Format (CGIF), eXtended Common Logic Markup Language (XCL).

document result of serializing an OMS or DOL library using a given serialization.

standoff markup way of providing annotations to subjects in external resources, without embedding them into the original resource (here: OMS).

abstract syntax
parse tree term language for representing documents in a machine-processable way
NOTE An abstract syntax can be specified as a MOF metamodel **NR25**. Then abstract abstract syntax documents can be represented as XMI **NR27** documents.

4.9 Semantics

formalization precise mathematical entity capturing an informal or semi-formal entity.

formal semantics assignment of a mathematical meaning to a language by mapping the abstract syntax to suitable semantic domains.
NOTE A formal semantics is a formalization of the meaning of a language.

semantic domain mathematically-defined set of values that can represent the intended meanings of language constructs.

semantic rule specification of a mapping from expressions for some meta class in the abstract syntax to a semantic domain.

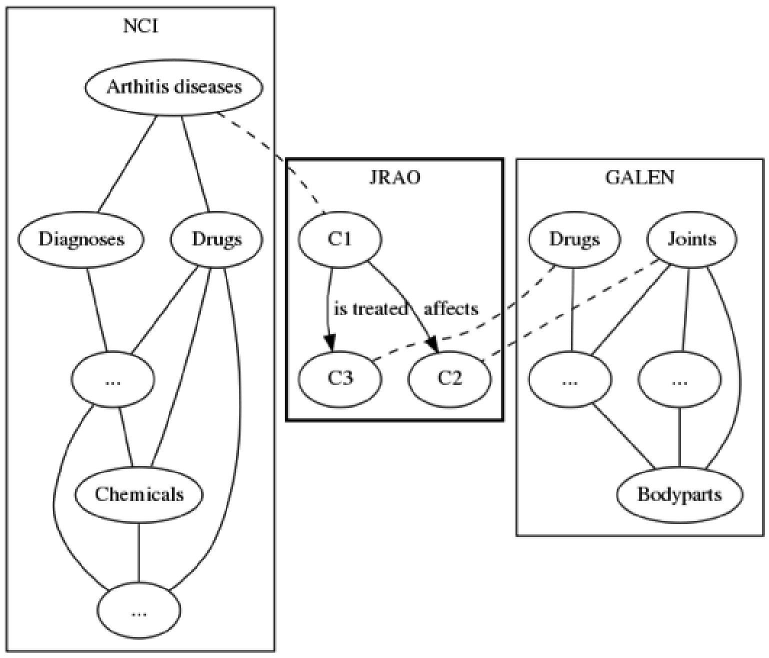

Figure 7.1 – JRAO – Example for Module Extraction

7.5 Use Case Onto-4: Interoperability Between Closed-World Data and Open-World Metadata

Data collection has become easier and much more widespread over the years. This data has to be assigned a meaning somehow, which occurs traditionally in the form of metadata annotations. For instance, consider geographical datasets derived from satellite data and raw sensor readings. Current implementations in, e.g., ecological economics [2] require manual annotation of datasets with the information relevant for their processes. While there have been attempts to standardize such information [11], metadata for datasets of simulation results are more difficult to standardize. Moreover, it is resource-consuming to link the data to the metadata, to ensure the metadata itself is of good quality and consistent, and to actually exploit the metadata when querying the data for data analysis.

The data is usually represented in a database or RDF triple store, which work with a closed world assumption on the dataset, and are not expressive enough to incorporate the metadata 'background knowledge', such as the conditions for validity of the physical laws in the ~~MDE~~ model of the object of observation. These metadata require a more expressive language, such as OWL or Common Logic, which operate under an open-world semantics. However, it is unfeasible to translate the whole large dataset into OWL or first-order logic. To 'meet in the middle', it is possible to declare bridge rules (i.e., a mapping layer) that can link the metadata to the data. This approach can be used for intelligent data analysis that combines the data and metadata through querying the system. It enables the analysis of the data on the conceptual layer, instead of users having to learn the SQL/SPARQL query languages and how the data is stored. There are various tools and theories to realize this, which is collectively called Ontology-Based Data Access/Management, see also [5].

The languages for representing the metadata or ontology, for representing the bridge rules or mapping assertions, and for representing the data are different yet they need to be orchestrated and handled smoothly in the system, be this for data analytics for large enterprises, for formulating policies, or in silico biology in the sciences.

DOL provides the framework for expressing such bridge rules in a systematic way, maintaining these, and building tools for them.

goes through the two ports bankCom and atmCom linked by a connector. The communication protocol on this connector is captured with a *protocol state machine*, see Fig. 7.4(c). The protocol state machine fixes in which order the messages verify, verified, reenterPIN, and markInvalid between atm and bank may occur. Figure 7.4(d) provides structural information in form of interfaces specifying what is provided and required at the userCom port and the bankCom port of the atm instance. An interface is a set of operations that other ~~MDE~~ model elements have to implement. In our case, the interfaces are described in a *class model*. Its component type ATM is further enriched with the OCL constraint trialsNum <= 3, which refines its semantics requiring that trialsNum must not exceed three.

Finally, the dynamic behavior of the atm object is specified by the *behavioral state machine* shown in Fig. 7.4(e). The machine consists of five states including Idle, CardEntered, etc. Beginning in the initial Idle state, the user can *trigger* a state change by entering the card. This has the *effect* that the parameter c from the card event is assigned to the cardId in the atm object (parameter names are not shown on triggers). Entering a PIN triggers another transition to PINEntered. Then the ATM requests verification from the bank using its bankCom port. The transition to Verifying uses a *completion event*: No explicit trigger is declared and the machine autonomously creates such an event whenever a state is completed, i.e., all internal activities of the state are finished (in our example there are no such activities). If the interaction with the bank results in reenterPIN, and the *guard* trialsNum < 3 is true, the user can again enter a PIN.

The ATM example in Fig. 7.4 consists of five different UML models, which naturally form a network. Coherence of this network is expressed as its consistency. It is assumed that XMI **NR27** representations of the relevant UML models have been stored at `http://www.example.org/uml/` in a single xmi-file `http://www.example.org/uml/atm.xmi` that contains a `uml:Model` element for each UML model whose `xmi:id` has a prefix `xxx` followed by an underscore. `xxx` is determined as follows:

Figure	**xxx**	**diagram type**
Fig. 7.4(a)	sd	sequence diagram
Fig. 7.4(b)	cmp	composite structure diagram
Fig. 7.4(c)	psm	protocol state machine
Fig. 7.4(d)	cd	class diagram
Fig. 7.4(e)	stm	state machine

```
%prefix( :       <http://www.example.org/uml/>
         uml:    <http://www.uml.org/spec/UML/>
         log:    <http://purl.net/DOL/logics/> )%
                 %% descriptions of logics ...
library ATM

view cd2stm = cd refined to { atm hide along stm2cd} end
view cd2psm = cd refined to { psm hide along psm2cd} end
network ATM_network = %consistent
                       cd, stm, psm, cmp,
                       cd2stm, cd2psm, abstract_to_concrete_atm
end
entailment atm = ATM_network entails sd
network Some_refined_ATM_network = ... end
refinement r = ATM_network refined to Some_refined_ATM_network
entailment e = Some_refined_ATM_network entails ATM_network
```

Here, `abstract_to_concrete_atm` is defined in the next section, and `stm2cd` and `psm2cd` are suitable logic projections extracting the classes, attributes and operations from a (protocol) state machine, delivering a class model.

7.12 Use Case Model-2: Refinements Between UML Models of Different Types, and Their Reuse

A problem is a lack of reusability of refinements: Consider a controller for an elevator, which is specified with a UML protocol state machine, enriched with UML sequence models and OCL constraints. Assume further that this UML model is not directly implemented, but first refined to a UML behavior state machine (which then can be automatically or semi-automatically transformed into some implementation using standard UML tools). However, there is no standardized language to express, document and maintain the refinement relation itself (UML only allows very simple refinements, namely between

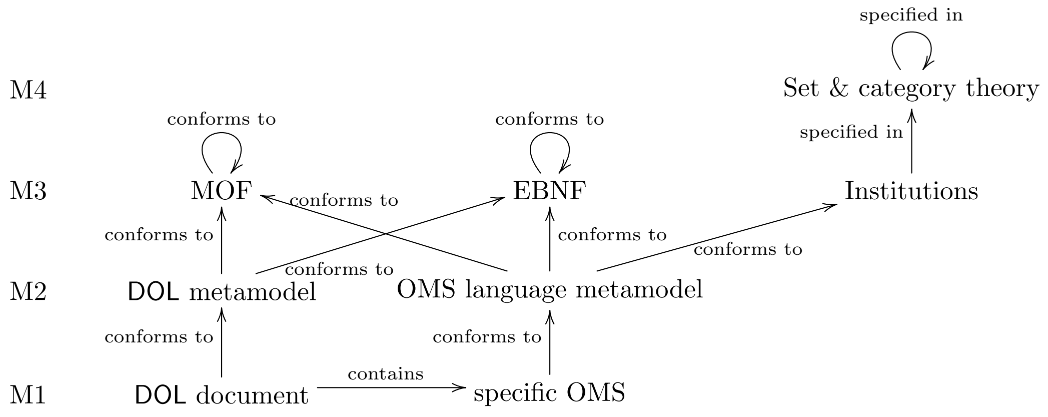

Figure 8.1 – DOL in the Metamodeling Hierarchy

8.6 Semantic Foundations of DOL

A large variety of OMS languages in use can be captured at an abstract level using the concept of *institutions* [18]. This allows the development of DOL independently of the particularities of a logical system and to use the notions of institution and logical language interchangeably. The main idea is to collect the non-logical symbols of the language in signatures and to assign to each signature the set of sentences that can be formed with its symbols. For each signature, DOL provides means for extracting the symbols it consists of, together with their kind. Institutions also provide a model theory, which introduces semantics for the language and gives a satisfaction relation between the ~~models~~ realizations and the sentences of a signature.

It is also possible to complement an institution with a proof theory, introducing a derivability relation between sentences, formalized as an *entailment system* [51]. In particular, this can be done for all logics that have so far been in use in DOL.

Since institutions allow the differences between OMS languages to be elided to common abstractions, the semantics of basic OMS is presented in a uniform way. The semantics of structured OMS, OMS mappings, OMS networks, and other DOL expressions is defined using model-theoretic constructions on top of institutions.

8.7 DOL Enables Expression of Logically Heterogeneous OMS and Literal Reuse of Existing OMS

DOL is a mechanism for expressing logically heterogeneous OMS. It can be used to combine sentences and structured OMS expressed in different conforming OMS languages and logics into single documents or modules. With DOL, sentences or structured OMS of previously existing OMS in conforming languages can be reused by literally including them into a DOL OMS. A minimum of wrapping constructs and other annotations (e.g., for identifying the language of a sentence) are provided. See the MOF metaclass `OMS` in clause 9.

A heterogeneous OMS can import several OMS expressed in different conforming logics, for which suitable translations have been defined in the logic graph provided in annex I or in an extension to it that has been provided when establishing the conformance of some other logic with DOL. Determining the semantics of the heterogeneous OMS requires a translation into a common target language to be applied (cf. clause 10). This translation is determined via a lookup in the transitive closure of the logic graph. Depending on the reasoners available in the given application setting, it can, however, be necessary to employ a different translation. Authors can express which one to employ. However, DOL provides default translations, which are applied unless the user specifies a translation that deviates from the default. Both default and non-default translations may be combined to multi-step translations.

— a ReductionOMS, applying a reduction (given by an Reduction) to an OMS (see use cases 7.9, 7.10 and 7.11 and Appendix M.9 and M.10 for examples);

— a ExtractionOMS, applying a module extraction operator (given by an Extraction) to an OMS (see use case 7.4 for an example);

— a QualifiedOMS, which is an OMS qualified with the OMS language that is used to express it.

Moreover, annex L informatively introduces Applications, which apply a substitution to an OMS.

A ConservativityStrength specifies additional relations that may hold between an OMS and its extension (or union with other OMS), like conservative or definitional extension. The rationale is that the extension should not have impact on the original OMS that is being extended.

An OMS definition OMSDefinition names an OMS. It can be optionally marked as inconsistent, consistent, monomorphic or having a unique ~~model~~ realization using ConservativityStrength. More precisely, 'consequence-conservative' here requires the OMS to have only tautologies as signature-free logical consequences, while 'notconsequence-conservative' expresses that this is not the case. 'model-conservative' requires satisfiability of the OMS, 'not-model-conservative' its unsatisfiability. 'definitional' expresses that the OMS has a unique ~~model~~ realization (see Appendix M.5 for an example); this may be interesting for characterizing OMS (e.g. returned by model finders) that are used to describe single ~~models~~realizations.

The DOL metamodel for extension OMS is shown in Fig. 9.6. ExtendingOMS is a subclass of OMS, containing those OMS that may be used to extend a given OMS within an ExtensionOMS. An ExtendingOMS can be one of the following:

— a basic OMS BasicOMS written inline, in a conforming serialization of a conforming OMS language (which is defined outside this standard; practically every example uses basic OMS)[16];

— a reference (through an IRI) to an OMS (OMSReference, many examples illustrate this); or

— a RelativeClosureOMS, applying a closure operator to a basic OMS or OMS reference (these two are hence joined into ClosableOMS). A closure forces the subsequently declared non-logical symbols to be interpreted in a minimal or maximal way, while the non-logical symbols declared in the local environment are fixed. Variants of closure are minimization, maximization, freeness (minimizing also data sets and equalities on these, which enables the inductive definition of relations and datatypes), and cofreeness (enabling the coinductive definition of relations and datatypes). See Annex M.6 for examples of the former two, and Annex M.11 for examples of the latter two.

Recall that the local environment is the OMS built from all previously-declared symbols and axioms.

Using ExtendingOMS, extensions of an OMS with an ExtendingOMS can be built. The latter can optionally be named and/or marked as conservative, monomorphic, definitional, weakly definitional or implied (using a ConservativityStrength, see clause 4.3 for details). Note that an ExtendingOMS used in an extension must not be an OMSReference.

Furthermore, OMS can be constructed using

— closures of an OMS with a Closure. This is similar to a RelativeClosureOMS, but the non-logical symbols to be minimized/maximized and to be varied are explicitly declared here (while a RelativeClosureOMS takes the local environment to be fixed, i.e. not varied);

— a translation OMSTranslation of an OMS into a different signature or OMS language. The former is done using a SymbolMap, specifying a map of symbols to symbols. The latter is done using an OMS language translation OMSLanguageTranslation can be either specified by its name, or be inferred as the default translation to a given target (the source will be inferred as the OMS language of the current OMS);

— a Reduction of an OMS to a smaller signature and/or less expressive logic (that is, some non-logical symbols and/or some parts of the ~~model structure~~ structure of the realization are hidden, but the semantic effect of sentences involving these is kept). The former is done using a SymbolList, which is a list of non-logical symbols that are to be hidden. The latter uses an OMSLanguageTranslation denoting a logic projection that is used as logic reduction to a less expressive OMS language.

[16] In this place, any OMS in a conforming serialization of a conforming OMS language is permitted. However, DOL's module sublanguage should be used instead of the module sublanguage of the respective conforming OMS language; e.g. DOL's OMS reference and extension construct should be preferred over OWL's import construct.

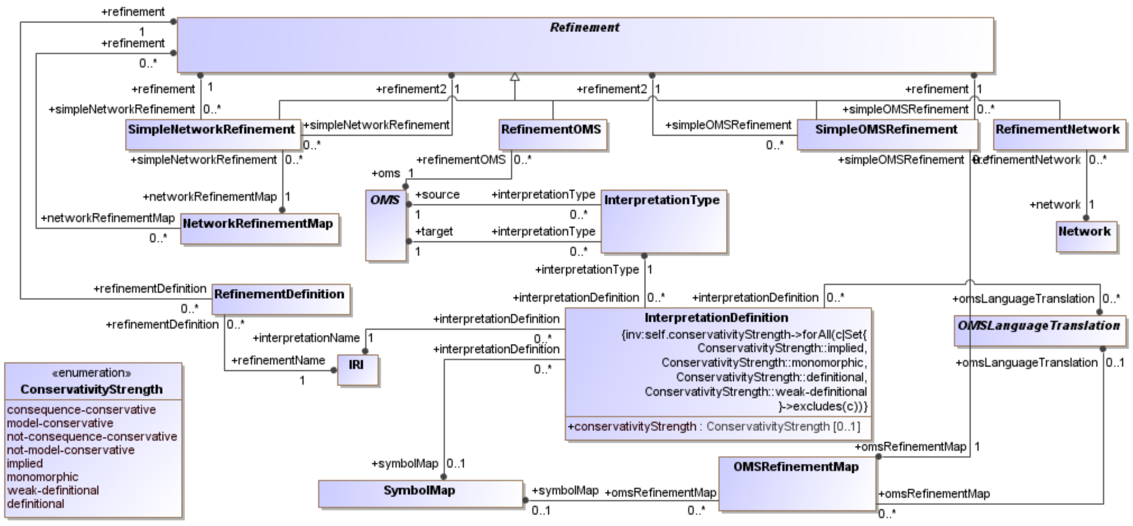

Figure 9.8 – DOL metamodel: Interpretations and refinements

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.
An `InterpretationDefinition` specifies source and target OMS (forming the `InterpretationType`), as well as a `SymbolMap` and/or an `OMSLanguageTranslation`. The `SymbolMap` in an interpretation always must lead to a signature morphism. A proof obligation expressing that the source OMS, when translated along the signature morphism and/or the `OMSLanguageTranslation`, logically follows from the target OMS.

A symbol map in an interpretation is **required** to cover all non-logical symbols of the source OMS; the semantics specification in clause 10 makes this assumption. (Mapping a non-logical symbol twice is an error. Mapping two source non-logical symbols to the same target non-logical symbol is legal, this is a non-injective OMS mapping.)

`Refinements` subsume interpretations (via `SimpleRefinements`), but allow the specification of much more complex relation between OMS (and OMS networks). The style differs from interpretation in that even a single OMS is a refinement (via `RefinementOMS`); this corresponds to the source of an interpretation. Using `SimpleOMSRefinements`, a refinement can be further specialized to a (target) OMS via an `OMSRefinementMap`. The latter involves a symbol map and/or OMS language translation, analogously to interpretations. With this style of notation, simple refinements can be easily chained up (which cannot be done using interpretations). Refinements themselves can also be refined, also by other refinements—this amounts to the possibility of composing refinements. Furthermore, refinements can also be specified between networks (`SimpleNetworkRefinement`, see use case 7.10 for an example). A refinement between OMS networks consists of a list of ordinary refinements (between OMS), one for each node in the source network (the OMS refinement is then required to refine the node in the source network to some node in the target network). The list may also include network refinements, much in the same way as network definitions also may include other networks. All the ordinary refinements occuring as components of the network refinement have to be compatible in a sense exemplified at the end of clause 7.10.

The DOL metamodel for entailments and equivalences is shown in Fig. 9.9. An entailment is a variant of an interpretation where all symbols are mapped identically, while an equivalence states that the ~~model classes of~~ classes of realizations of two OMS are in bijective correspondence. As for refinements, entailments and equivalences are also possible between networks (`NetworkNetworkEntailment` and `NetworkEquivalence`). An entailment between a network as premise and an OMS as conclusion (`NetworkOMSEntailment`) specifies that all ~~models~~ realizations of the network, when restricted to a given node (given by an IRI), are ~~models~~ realizations of the OMS.

The DOL metamodel for alignments is shown in Fig. 9.10. Signature morphisms used in interpretations and refinements use a functional style of mapping symbols of OMS. In contrast to this style, an alignment provides a relational connection

— *a category*[21] Sig *of* signatures *and* signature morphisms*,*

— *a functor* **Sen**: Sig $\longrightarrow \mathbb{S}et$ *giving, for each signature* Σ*, the set of* sentences $\mathbf{Sen}(\Sigma)$*, and for each signature morphism* $\sigma : \Sigma \to \Sigma'$*, the* sentence translation map $\mathbf{Sen}(\sigma) : \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$*, where often* $\mathbf{Sen}(\sigma)(\varphi)$ *is written as* $\sigma(\varphi)$*,*

— *a functor* $\mathbf{Mod} : \mathsf{Sig}^{op} \to \mathbb{C}at$ *giving, for each signature* Σ*, the category of* ~~models~~realizations[22] $\mathbf{Mod}(\Sigma)$*, and for each signature morphism* $\sigma \colon \Sigma \longrightarrow \Sigma'$*, the* reduct functor $\mathbf{Mod}(\sigma) : \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$*, where often* $\mathbf{Mod}(\sigma)(M')$ *is written as* $M'|_\sigma$*, and* $M'|_\sigma$ *is called the* σ-reduct *of* M'*, while* M' *is called a* σ-expansion *of* $M'|_\sigma$*,*

— *a satisfaction relation* $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ *for each* $\Sigma \in |\mathsf{Sig}|$*,*

such that for each $\sigma \colon \Sigma \longrightarrow \Sigma'$ *in* Sig *the following* ***satisfaction condition*** *holds:*

$$M' \models_{\Sigma'} \sigma(\varphi) \iff M'|_\sigma \models_\Sigma \varphi \qquad (\star)$$

for each $M' \in |\mathbf{Mod}(\Sigma')|$ *and* $\varphi \in \mathbf{Sen}(\Sigma)$*, expressing that truth is invariant under change of notation and context.* □

Definition 3 (Propositional Logic) *The signatures of propositional logic are sets* Σ *of propositional symbols, and signature morphisms are just functions* $\sigma : \Sigma_1 \to \Sigma_2$ *between these sets. A* Σ*~~-model~~ -realization is a function* $M : \Sigma \to \{True, False\}$*, and the reduct of a* Σ_2*~~-model~~ -realization* M_2 *along a signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$ *is the* Σ_1*~~-model~~ -realization given by the composition of* σ *with* M_2*.* Σ*-sentences are built from the propositional symbols with the usual connectives, and sentence translation is replacing the propositional symbols in* Σ *along the morphism. Finally, the satisfaction relation is defined by the standard truth-tables semantics. It is straightforward to see that the satisfaction condition holds.* □

Definition 4 (Common Logic — CL) *A common logic signature* Σ *(called vocabulary in Common Logic terminology) consists of a set of names, with a subset called the set of discourse names, and a set of sequence markers. A* Σ*~~-model~~ -realization consists of a set UR, the universe of reference, with a non-empty subset* $UD \subseteq UR$*, the universe of discourse, and four mappings:*

— *rel from UR to subsets of* $UD^* = \{\langle x_1, \ldots, x_n \rangle \mid x_1, \ldots, x_n \in UD\}$ *(i.e., the set of finite sequences of elements of UD);*

— *fun from UR to total functions from* UD^* *into UD;*

— *int from names in* Σ *to UR, such that* $int(v)$ *is in UD if and only if* v *is a discourse name;*

— *seq from sequence markers in* Σ *to* UD^**.*

A Σ*-sentence is a first-order sentence, where predications and function applications are written in a higher-order like syntax:* $t(s)$*. Here,* t *is an arbitrary term, and* s *is a sequence term, which can be a sequence of terms* $t_1 \ldots t_n$*, or a sequence marker. A predication* $t(s)$ *is interpreted by evaluating the term* t*, mapping it to a relation using rel, and then asking whether the sequence given by the interpretation* s *is in this relation. Similarly, a function application* $t(s)$ *is interpreted using fun. Otherwise, interpretation of terms and formulae is as in first-order logic. A difference to first-order logic is the presence of sequence terms (namely sequence markers and juxtapositions of terms), which denote sequences in* UD^**, with term juxtaposition interpreted by sequence concatenation. Note that sequences are essentially a non-first-order feature that can be expressed in second-order logic. For details, see [30].*

A CL *signature morphism consists of two maps between the sets of names and of sequence markers, such that the property of being a discourse name is preserved and reflected.*[23] *~~Model reducts~~ Reducts leave UR, UD, rel and fun untouched, while int and seq are composed with the appropriate signature morphism component.* □

Further examples of institutions are: $\mathcal{SROIQ}(D)$, unsorted first-order logic, many-sorted first-order logic, and many others. Note that the reduct of a ~~model~~ realization is generally given by forgetting some of its parts.

For the rest of the section, an arbitrary institution is considered.

Definition 5 (Theory) *A* ***theory*** *is a pair* (Σ, Δ) *where* Σ *is a signature and* Δ *is a set of* Σ*-sentences.*

[21] See [1][47] for an introduction into category theory.

[22] To avoid confusion with models in the sense of model-driven engineering, we use ‘realization‘ instead of the commonly used term ‘model‘ in logic.

[23] That is, a name is a discourse name if and only if its image under the signature morphism is.

Given a theory $T = (\Sigma, \Delta)$, the class of T-~~models~~ realization is the class of all Σ-~~models~~ realizations M such that $M \models \delta$, for each sentence $\delta \in \Delta$. A theory (Σ, Δ) is **consistent** if at least one ~~Σ-model~~ (Σ, Δ)-realization exists. **Semantic entailment** is defined as usual: for a theory ~~$\Delta \subseteq \mathbf{Sen}(\Sigma)$~~ (Σ, Δ) and $\varphi \in \mathbf{Sen}(\Sigma)$, Δ entails φ, written $\Delta \models \varphi$, if all ~~models~~ realizations satisfying all sentences in Δ also satisfy φ. For a theory (Σ, Δ), we write $\Delta^\bullet$ for the set of all Σ-sentences φ such that $\Delta \models \varphi$.

Definition 6 (Theory morphism) *A **theory morphism** $\phi : (\Sigma, \Delta) \to (\Sigma', \Delta')$ is a signature morphism $\phi : \Sigma \to \Sigma'$ such that $\Delta' \models \phi(\Delta)$.*

Institution comorphisms capture the intuition of encoding or embedding a logic into a more expressive one.

Definition 7 (Institution Comorphism) *An **institution comorphism** from an institution $I = (\mathsf{Sig}^I, \mathsf{Mod}^I, \mathsf{Sen}^I, \models^I)$ to an institution $J = (\mathsf{Sig}^J, \mathsf{Mod}^J, \mathsf{Sen}^J, \models^J)$ consists of a functor $\Phi : \mathsf{Sig}^I \longrightarrow \mathsf{Sig}^J$, and two natural transformations $\beta : \mathsf{Mod}^J \circ \Phi^{op} \Longrightarrow \mathsf{Mod}^I$ and $\alpha : \mathsf{Sen}^I \Longrightarrow \mathsf{Sen}^J \circ \Phi$, such that for each I-signature Σ, each sentence $\varphi \in \mathsf{Sen}^I(\Sigma)$ and each ~~model~~ realization $M' \in |\mathsf{Mod}^J(\Phi(\Sigma))|$*

$$M' \models^J_{\Phi(\Sigma)} \alpha_\Sigma(\varphi) \iff \beta_\Sigma(M') \models^I_\Sigma \varphi.$$

*holds, called the **satisfaction condition**.* $\square$

Here, $\Phi(\Sigma)$ is the translation of the signature Σ from institution I to institution J, $\alpha_\Sigma(\varphi)$ is the translation of the Σ-sentence φ to a $\Phi(\Sigma)$-sentence, and $\beta_\Sigma(M')$ is the translation (or perhaps better: reduction) of the $\Phi(\Sigma)$-~~model~~ realization M' to a Σ-~~model~~ realization. Naturality of α and β means that for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ in I the following squares commute:

$$\begin{tikzcd}
\mathsf{Sen}^I(\Sigma_1) \arrow[r, "\alpha_{\Sigma_1}"] \arrow[d, "\mathsf{Sen}^I(\sigma)"'] & \mathsf{Sen}^J(\Phi(\Sigma_1)) \arrow[d, "\mathsf{Sen}^J(\Phi(\sigma))"] & \mathsf{Mod}^J(\Phi(\Sigma_2)) \arrow[r, "\beta_{\Sigma_2}"] \arrow[d, "\mathsf{Mod}^J(\Phi(\sigma))"] & \mathsf{Mod}^I(\Sigma_2) \arrow[d, "\mathsf{Mod}^I(\sigma)"] \\
\mathsf{Sen}^I(\Sigma_2) \arrow[r, "\alpha_{\Sigma_2}"'] & \mathsf{Sen}^J(\Phi(\Sigma_2)) & \mathsf{Mod}^J(\Phi(\Sigma_1)) \arrow[r, "\beta_{\Sigma_1}"'] & \mathsf{Mod}^I(\Sigma_1)
\end{tikzcd}$$

A comorphism is:

— *faithful* if logical consequence is preserved and reflected along the comorphism:

$$\Gamma \models^I \varphi \iff \alpha(\Gamma) \models^J \alpha(\varphi)$$

— *model-expansive* if each β_Σ is surjective;

— *(weakly) exact* if for each signature morphism $\sigma\colon \Sigma_1 \longrightarrow \Sigma_2$, the naturality diagram

$$\begin{tikzcd}
\mathsf{Mod}^J(\Phi(\Sigma_2)) \arrow[r, "\beta_{\Sigma_2}"] \arrow[d, "\mathsf{Mod}^J(\Phi(\sigma))"] & \mathsf{Mod}^I(\Sigma_2) \arrow[d, "\mathsf{Mod}^I(\sigma)"] \\
\mathsf{Mod}^J(\Phi(\Sigma_1)) \arrow[r, "\beta_{\Sigma_1}"'] & \mathsf{Mod}^I(\Sigma_1)
\end{tikzcd}$$

admits (weak) amalgamation, i.e. any for any two ~~models~~ realizations $M_2 \in |\mathsf{Mod}^I(\Sigma_2)|$ and $M_1' \in |\mathsf{Mod}^J(\Phi(\Sigma_1))|$ with $M_2|_\sigma = \beta_{\Sigma_1}(M_1')$, there is a unique (not necessarily unique) $M_2' \in |\mathsf{Mod}^J(\Phi(\Sigma_2))|$ with $\beta_{\Sigma_2}(M_2') = M_2$ and $M_2'|_{\Phi(\sigma)} = M_1'$;

— a *subinstitution comorphism* if Φ is an embedding, each α_Σ is injective and each β_Σ is bijective[24];

— an *inclusion comorphism* if Φ and each α_Σ are inclusions, and each β_Σ is the identity.

It is known that each subinstitution comorphism is model-expansive and each model-expansive comorphism is also faithful. Faithfulness means that a proof goal $\Gamma \models^I \varphi$ in I can be solved by a theorem prover for J by just feeding the theorem prover with $\alpha(\Gamma) \models^J \alpha(\varphi)$. Subinstitution comorphism preserve the semantics of more advanced DOL structuring constructs such as OMS translation and OMS reduction.

[24] An isomorphism if ~~model~~ morphisms of realizations are taken into account.

Definition 8 *Given an institution $I = (\mathsf{Sig}^I, \mathsf{Mod}^I, \mathsf{Sen}^I, \models^I)$, the institution of its theories, denoted I^{th}, can be defined as follows. The category of signatures of I^{th} is the category of I-theories and I-theory morphisms, denoted Th^I. For each theory (Σ, Δ), its sentences are just Σ-sentences in I, and its ~~models~~ realizations are just Σ-~~models~~ -realizations in I that satisfy the sentences in Δ, while the (Σ, Δ)-satisfaction is the Σ-satisfaction of sentences in ~~models~~ realizations of I.* □

Using this notion, logic translations can be defined that include axiomatization of parts of the syntax of the source logic into the target logic.

Definition 9 *Let $I = (\mathsf{Sig}^I, \mathsf{Mod}^I, \mathsf{Sen}^I, \models^I)$ and $J = (\mathsf{Sig}^J, \mathsf{Mod}^J, \mathsf{Sen}^J, \models^J)$ be two institutions. A **theoroidal institution comorphism** from I to J is a institution comorphism from I to J^{th}.* □

Institution morphisms capture the intuition of projecting from a more expressive logic to a less expressive one.

Definition 10 (Institution Morphism) *An **institution morphism** from an institution $I = (\mathsf{Sig}^I, \mathsf{Mod}^I, \mathsf{Sen}^I, \models^I)$ to an institution $J = (\mathsf{Sig}^J, \mathsf{Mod}^J, \mathsf{Sen}^J, \models^J)$ consists of a functor $\Phi : \mathsf{Sig}^I \longrightarrow \mathsf{Sig}^J$, and two natural transformations $\beta : \mathsf{Mod}^I \Longrightarrow \mathsf{Mod}^J \circ \Phi^{op}$ and $\alpha : \mathsf{Sen}^J \circ \Phi \Longrightarrow \mathsf{Sen}^I$, such that for each I-signature Σ, each sentence $\varphi \in \mathsf{Sen}^J(\Phi(\Sigma))$ and each ~~model~~ realization $M \in \mathsf{Mod}^I(\Sigma)$*

$$M \models^I_\Sigma \alpha_\Sigma(\varphi) \iff \beta_\Sigma(M) \models^J_{\Phi(\Sigma)} \varphi.$$

*holds, called the **satisfaction condition**.* □

Colimits are a categorical concept providing means of combining objects interconnected by morphisms, where the colimit glues together objects along the morphisms. They can be employed for constructing larger theories from already available smaller ones, see [18]. For a formal mathematical definition, see P.1.1.

A major property of colimits of specifications is *amalgamation* (also related to 'exactness' [14]). It can be intuitively explained as stating that ~~models~~ realizations of given specifications can be combined to yield a uniquely determined ~~model~~ realization of a colimit specification, provided that the original ~~models~~ realizations coincide on common components. Amalgamation is a common technical assumption in the study of specification semantics [67].

In the following, fix an arbitrary institution $I = (\mathsf{Sig}, \mathsf{Sen}, \mathsf{Mod}, \models)$.

Definition 11 *Given a network $D\colon J \longrightarrow \mathsf{Sig}^I$, a family of ~~models~~ realizations $\mathcal{M} = \{M_p\}_{j\in|J|}$ is* consistent with *D (or sometimes compatible with D) if for each node p of D, $M_p \in Mod(D(p))$ and for each edge $e : p \to q$, $M_p = M_q|_{D(e)}$. A cocone $(\Sigma, (\mu_j)_{j\in|J|})$ over the network $D\colon J \longrightarrow \mathsf{Sig}^I$ is called* weakly amalgamable *if it is mapped to a weak limit by* Mod*. For ~~models~~realizations, this means that for each D-compatible family of ~~models~~ realizations $(M_j)_{j\in|J|}$, there is a Σ-~~model~~ -realization M, called an amalgamation of $(M_j)_{j\in|J|}$, with $M|_{\mu_j} = M_j$ $(j \in |J|)$, and similarly for ~~model morphisms~~ morphisms of realizations. If this ~~model~~ realization is unique, the cocone is called* amalgamable*. I (or* Mod*) admits* (finite) (weak) amalgamation *if (finite) colimit cocones are (weakly) amalgamable. Finally, I is called* (weakly) semi-amalgamable *if it has pushouts and admits (weak) amalgamation for these.* □

[8] studies conditions for existence of weakly amalgamable cocones in a heterogeneous setting, where the network consists of signatures (or theories) in different logics. Since a network may admit more than one weakly amalgamable cocone, a selection operation is required both for the weakly amalgamable cocone of a network and for the (potentially non-unique) amalgamation of a family of ~~models~~ realizations compatible with the network. This allows us to define a function *colimit* taking as argument a network of heterogeneous signatures and returning the selected weakly amalgamable cocone for the network and a function $\oplus$ taking as argument a family of ~~models~~ realizations compatible with a network and returning its selected amalgamation.

10.3 Semantics of DOL Language Constructs

The semantics of DOL is based on a fixed (but in principle arbitrary) heterogeneous logical environment. The semantic domains are based on this heterogeneous logical environment. A specific heterogeneous logical environment is given in the annexes.

A heterogeneous logical environment is given by a collection of OMS languages and OMS language translations[25)], a collection of institutions, institution morphisms and institution comorphisms (serving as logics, logic reductions and logic translations), and a collection of serializations. Moreover, some of the institution comorphisms are marked as default translations (but only at most one between a given source and target institution), and there is a binary relation *supports* between OMS languages and institutions, and a binary relation *supports* between OMS languages and serializations. Each language is required to have a default logic and serialization. Moreover, we assume that institutions, institution morphisms and institution comorphisms are uniquely identified by names, and we use the notation $\Gamma(n)$ for the institution, institution morphism and institution comorphism identified by the name n int the heterogeneous logical environment Γ.

We are going to require existence of union and difference operations on the signatures of an institution in the heterogeneous logical environment. These concepts could be captured in a categorical setting using *inclusion systems* [14]. However, inclusion systems are too strong for the purposes of this specification. Therefore, weaker assumptions will be used.

Definition 12 *An* inclusive category *[20] is a category having a broad subcategory*[26)] *which is a partially ordered class with a least element (denoted $\emptyset$), finite products and coproducts, called intersection (denoted $\cap$) and union (denoted $\cup$) such that for each pair of objects A, B, $A \cup B$ is a pushout of $A \cap B$ in the category.* □

A category *has pushouts which preserve inclusions* iff there exists a pushout

$$\begin{tikzcd} A \arrow[r, hook] \arrow[d] & A' \arrow[d] \\ B \arrow[r, hook] & B' \end{tikzcd}$$

for each span where one arrow is an inclusion.

A functor between two inclusive categories is inclusive if it takes inclusions in the source category to inclusions in the target category.

Definition 13 *An institution is* weakly inclusive *if*

— Sig *is inclusive and has pushouts which preserve inclusions,*
— Sen *is inclusive, and*
— *each ~~model category~~ category of realizations has a broad subcategory of inclusions.* □

Let I be a weakly inclusive institution. I *has differences*, if there is a binary operation $\setminus$ on signatures, such that for each pair of signatures Σ_1, Σ_2, the greatest signature Σ such that

1) $\Sigma \subseteq \Sigma_1$
2) $\Sigma \cap \Sigma_2 = \emptyset$

exists and is equal to $\Sigma_1 \setminus \Sigma_2$.

We will write $\iota_{A \subseteq B}$ for the inclusion of A in B in an inclusive category, when such an inclusion exists. If $\mathcal{I}$ is an inclusive institution and $\Sigma \subseteq \Sigma'$ is an inclusion of signatures, we write $M'|_\Sigma$ for the reduct of a Σ'-~~model~~ realization M' along the inclusion $\iota_{\Sigma \subseteq \Sigma'}$.

To be able to talk about the symbols of a signature in a formal way, it is required that the category of signatures of an institution is an inclusive category with symbols, as defined below:

Definition 14 *An* inclusive category with symbols *is an inclusive category $\mathbb{C}$ equipped with a faithful functor $|\_| : \mathbb{C} \to \mathbb{S}et$*[27)] *that preserves inclusions.* □

[25)]The terms *OMS language* and *serialization* are not defined formally. For this semantics, it suffices to know that there is a language-specific semantics of basic OMS as defined below.

[26)]That is, with the same objects as the original category.

[27)]That is, $(\mathbb{C}, |\_|)$ is a concrete category.

Moreover, if $\sigma : \Sigma \to \Sigma'$ is a signature morphism, it uniquely determines a map $|\sigma| : |\Sigma| \to |\Sigma'|$.

After these preliminaries, we can now list the assumptions made about the institutions in a heterogeneous logical environment. It is required that for each institution in the heterogeneous logical environment there is a trivial signature $\emptyset$ with ~~model class~~ class of realizations $\mathcal{M}_\emptyset$ and such that there exists a unique signature morphism from $\emptyset$ to any signature of the institution. Moreover, the existence of a partial union operation on institutions is required, denoted $\bigcup$: $L_1 \bigcup L_2 = (L, \rho_1 : L_1 \to L, \rho_2 : L_2 \to L)$, when defined, where L is an institution and ρ_1 and ρ_2 are institution comorphisms, giving the embedding of L_1 and respectively L_2 in L. Finally, some of the comorphisms are marked as default translations and some of the morphisms as default projections, with the condition that between any two institutions at most one comorphism and at most one morphism is marked as default.

For each institution $\mathcal{I}$ in the heterogeneous logical environment, it is further required that there is:

— a function giving the semantics of a basic OMS. It has the format

$$semBasic_{(lang,logic,ser)}(\Sigma, O) = (\Sigma', \Delta')$$

where O is a `BasicOMS`, Σ gives the context of previous declarations, Σ' is the resulting signature and Δ' is the resulting set of sentences. It is required then that $\Sigma \subseteq \Sigma'$.

— a function $makeMorphism_\mathcal{I}$ that turns symbol maps into signature morphisms,

— a function $sameName_\mathcal{I}$ that takes as arguments two signatures Σ_1 and Σ_2 of $\mathcal{I}$ and returns as result the list of all pairs of symbols (s_i^1, s_i^2) such that $s_i^1 \in |\Sigma_1|$ and $s_i^2 \in |\Sigma_2|$ and the symbols have the same name. The relation represented by $sameName_\mathcal{I}(\Sigma_1, \Sigma_2)$ must be an equivalence relation.

— a relativization function $relativize_\mathcal{I}$ taking as argument a theory and giving as result a theory, and a function $theoryOfCorrespondences$ for translating correspondences of alignments into sentences in the logic according to a given assumption about the semantics of the alignment, both needed in Section 10.3.4.

Further, for each institution, it is required that there exist union and difference operations on signatures.

DOL follows a model-theoretic approach on semantics: the semantics of OMS will be defined as a class of ~~models~~ realizations over some signature of an institution. This is called *model-level* semantics. In some cases, but not in all, one can also define a *theory-level* semantics of an OMS as a set of sentences over some signature of an institution. The two semantics are related by the fact that, when both the model-level and the theory-level semantics of an OMS are defined, they are compatible in the sense that the class of ~~models~~ realizations given by the model-level semantics is exactly the ~~model class of~~ class of realizations of the theory given by the theory-level semantics.

The following unifying notation is used for the two semantics of an OMS O:

— the institution of O is denoted $\mathbf{Inst}(O)$,

— the signature of O is denoted $\mathsf{Sig}(O)$ (which is a signature in $\mathbf{Inst}(O)$),

— the class of models of O is denoted $\mathsf{Mod}(O)$ (which is a class of ~~models~~ realizations over $\mathsf{Sig}(O)$),

— the set of axioms of O is denoted $\mathsf{Th}(O)$ (which is a set of sentences over $\mathsf{Sig}(O)$).

Moreover, the semantics of O is the tuple $sem(O) = (I, \Sigma, \mathcal{M}, \Delta)$ where $\mathbf{Inst}(O) = I$, $\mathsf{Sig}(O) = \Sigma$, $\mathsf{Mod}(O) = \mathcal{M}$ and $\mathsf{Th}(O) = \Delta$. In the following, we will freely mix these two equivalent descriptions of the semantics. That is, whenever $sem(O)$ is determined in some the context, then also its components $\mathbf{Inst}(O)$, $\mathsf{Sig}(O)$, $\mathsf{Mod}(O)$ and $\mathsf{Th}(O)$ are determined. Vice versa, if the four components are determined, then so is $sem(O)$.

The theory-level semantics of O can be undefined, and then so is $\mathsf{Th}(O)$. When $\mathsf{Th}(O)$ is defined, $\mathsf{Mod}(O)$ can be obtained as $\mathsf{Mod}(O) = \{M \in \mathsf{Mod}(\mathsf{Sig}(O)) \mid M \models \mathsf{Th}(O)\}$.

Intuitively, OMS mappings denote various types of links between two or more OMS. The semantics of OMS mappings can be captured uniformly as a graph whose nodes N are labeled with

— $\mathbf{Name}(N)$, the name of the node

— $\mathbf{Inst}(N)$, the institution of the node

- $\mathsf{Sig}(N)$, the signature of the node
- $\mathsf{Mod}(N)$, the class of $\mathsf{Sig}(N)$-~~models~~ realizations of the node
- $\mathsf{Th}(N)$, the set of $\mathsf{Sig}(N)$-sentences of the node

and which has two kinds of edges:

- import links (written using single arrows, $S \to T$)
- theorem links (written using double arrows, $S \Rightarrow T$)

both labeled with heterogeneous signature morphisms between the signatures of the source and target nodes (i.e. an edge from the node S to the node T is labeled with a pair (ρ, σ) where $\rho = (\Phi, \alpha, \beta) : \mathbf{Inst}(S) \to \mathbf{Inst}(T)$ is an institution comorphism and $\sigma : \Phi(\mathsf{Sig}(S)) \to \mathsf{Sig}(T)$ is a signature morphism in $\mathbf{Inst}(T)$). The theory of a node may be undefined, as in the case of OMS, and when it is defined, the class of ~~models~~ realizations of that node is the class of models of $\mathsf{Th}(N)$. For brevity, the label of a node may be written as a tuple. Further, it is required that any OMS can be assigned a unique name.

The semantics of a network of OMS is a graph whose nodes are labeled like in the semantics of OMS mappings and edges are labeled with heterogeneous signature morphisms. The intuition is that network provide means of putting together graphs of OMS and OMS mappings and of removing sub-graphs of existing networks.

The semantics of OMS generally depends on a global environment Γ containing:

- a graph of imports between OMS, as in the semantics of OMS mappings but only with import links between nodes, denoted $\Gamma.\mathit{imports}$
- a mapping from `IRI`s to semantics of OMS, OMS mappings, and OMS networks, that is also denoted by Γ, providing access to previous definitions,
- a prefix map, denoted $\Gamma.\mathit{prefix}$, that stores the declared prefixes,
- a triple $\Gamma.\mathit{current}$ that stores the current language, logic and serialization.

If Γ is such a global environment, $\Gamma[\mathtt{IRI} \mapsto \mathcal{S}]$ extends the domain of Γ with `IRI` and the newly added value of `IRI` in Γ is the semantic entity $\mathcal{S}$. $\Gamma_\emptyset$ is the empty global environment, i.e. the domain of $\Gamma_\emptyset$ is the empty set, its import graph $\Gamma.\mathit{imports}$ is empty, the prefix map is empty and the current triple contains the error logic together with its language and serialization. The union of two global environments Γ_1 and Γ_2, denoted $\Gamma_1 \cup \Gamma_2$, is defined only if the domains of Γ_1 and Γ_2, and of $\Gamma_1.\mathit{prefix}$ and $\Gamma_2.\mathit{prefix}$ are disjoint, and then $\Gamma_1 \cup \Gamma_2(\mathtt{IRI}) = \begin{cases} \Gamma_1(\mathtt{IRI}) & \text{if } \mathtt{IRI} \in \mathit{dom}(\Gamma_1) \\ \Gamma_2(\mathtt{IRI}) & \text{if } \mathtt{IRI} \in \mathit{dom}(\Gamma_2) \end{cases}$, $\Gamma_1 \cup \Gamma_2.\mathit{imports} = \Gamma_1.\mathit{imports} \cup \Gamma_2.\mathit{imports}$, $\Gamma_1 \cup \Gamma_2.\mathit{current} = \Gamma_1.\mathit{current}$ and $\Gamma_1 \cup \Gamma_2.\mathit{prefix} = \Gamma_1.\mathit{prefix} \cup \Gamma_2.\mathit{prefix}$. $\Gamma.\{\mathit{prefix} = \mathit{PMap}\}$ represents the global environment that sets the prefix map of Γ to PMap and $\Gamma.\{\mathit{current} = (\mathit{lang}, \mathit{logic}, \mathit{ser})\}$ is used for updating the current triple of Γ to $(\mathit{lang}, \mathit{logic}, \mathit{ser})$.

DOL assumes a *language-specific semantics* of native structured OMS, inherited from the OMS language. For a native document D in a language L, logic L' and serialization S, $\mathit{semNative}_{(L,L',S)}(D)$ denotes the language-specific semantics of D.

10.3.1 Semantics of Documents

In this section the semantics of DOL constructs regarding documents and DOL libraries is defined.

$$\boxed{\begin{aligned} \mathit{sem}(\mathtt{Document}) &= \Gamma \\ &: \mathit{LogicalEnvironment} \end{aligned}}$$

A document is either a DOL library, or a native document written in one of the languages supported by the heterogeneous logical environment.

For a `NativeDocument` *nativeDocument*,

$$\mathit{sem}(\mathit{nativeDocument}) = \Gamma''$$

10.3.2 Semantics of Networks

The semantics of networks of OMS is given with the help of a directed graph. Its nodes and edges are specified by the `NetworkElements`, which can be OMS, OMS mappings, or OMS networks. Intuitively, the graph of a network consists of the union of all graphs of the network elements it contains, where an OMS yields a graph with one isolated node. By convention, all imports in the graph $\Gamma.imports$ of the current context between nodes that are specified in the list of `NetworkElements` are also included in the graph of the network. The nodes and edges given in the `ExcludeExtensions` list are then removed from the graph of the network.

An additional `Id` can be specified for each node, with the purpose of letting the user specify a prefix in the colimit of a network for the symbols with the origin in that node that must be disambiguated.

The following auxiliary functions are used:

- $insert(G, \Gamma, iri, id)$, where G is a graph, Γ is a global environment, iri is an `IRI` and id is an `Id`, defined as follows:
  - if iri denotes an OMS in Γ, then a new node named iri and labeled with $\Gamma(iri)$ and with id is added to G, unless a node named iri already exists in G, and in this case G is left unchanged,
  - if iri denotes an OMS mapping or a network in Γ, the result is the union of G with the graph of $\Gamma(iri)$.
- $removeElement(\Gamma, G, anIRI)$, where G is a graph, Γ is a global environment and $anIRI$ is an `IRI`, defined as follows:
  - if $anIRI$ denotes an OMS in Γ, then the node labeled with $anIRI$ and all its incoming and outgoing edges are removed from G,
  - if $anIRI$ denotes an OMS mapping in Γ, then $\Gamma(anIRI)$ gives a graph G' and two nodes N_1 and N_2. Then all nodes of G' other than N_1 and N_2 and all the edges of G' are removed from G.
  - if $anIRI$ is a network in Γ, then all the nodes of its graph and all their incoming and outgoing edges are removed from G.
- $removePaths(\Gamma, G, iri_1, iri_2)$, where G is a graph, Γ is a global environment and iri_1, iri_2 are `IRI`s, whose result is that all paths of imports in G between the nodes labeled with iri_1 and iri_2 are removed from G.

Finally, the operation $addImports(\Gamma, G, [iri_1, \ldots, iri_n])$ adds to G all import edges in $\Gamma.imports$ between nodes which appear in the subgraph determined by $\Gamma(iri_1), \ldots, \Gamma(iri_n)$.

10.3.2.1 Semantics of network definitions

$$\boxed{\begin{aligned} sem(\Gamma, \texttt{NetworkDefinition}) &= \Gamma' \\ &: LogicalEnvironment \end{aligned}}$$

If n is a `NetworkDefinition`,

$$sem(\Gamma, n) = \Gamma'$$

where $\Gamma' = \Gamma[n.\texttt{networkName} \mapsto sem(\Gamma, n.\texttt{network})]$.

If n.`ConservativityStrength` is `model-conservative`, the semantics is only defined if the class of families of ~~models~~ realizations compatible with the graph $sem(\Gamma, n.\texttt{network})$ is not empty.

If n.`ConservativityStrength` is `consequence-conservative`, the semantics is defined only if all signature-free sentences that follow from the network, see entailment of OMS by networks, are tautologies.

If n.`ConservativityStrength` is `monomorphic`, the semantics is only defined if the class of families of ~~models~~ realizations compatible with the graph $sem(\Gamma, n.\texttt{network})$ consist of exactly one isomorphism class of families of ~~models~~realizations.

If n.`ConservativityStrength` is `weak-definitional`, the semantics is only defined if the class of families of ~~models~~ realizations compatible with the graph $sem(\Gamma, n.\texttt{network})$ is at most a singleton.

If n.`ConservativityStrength` is `definitional`, the semantics is only defined if the class of families of ~~models~~ realizations compatible with the graph $sem(\Gamma, n.\mathtt{network})$ is a singleton.

If n.`ConservativityStrength` is `not-model-conservative`, the semantics is only defined if the class of families of ~~models~~ realizations compatible with the graph $sem(\Gamma, n.\mathtt{network})$ is the empty set.

If n.`ConservativityStrength` is `not-consequence-conservative`, the semantics is defined only if not all signature-free sentences that follow from the network, see entailment of OMS by networks, are tautologies.

10.3.2.2 Semantics of networks

$$\boxed{\begin{aligned} sem(\Gamma, \mathtt{Network}) &= G \\ &: OMSGraph \end{aligned}}$$

If n is a network,

$$sem(\Gamma, n) = G'$$

where $sem(\Gamma, n.\mathtt{networkElement}) = G$ and $sem(\Gamma, G, n.\mathtt{excludedElement}) = G'$.

10.3.2.3 Semantics of sets of network elements

$$\boxed{\begin{aligned} sem(\Gamma, Set(\mathtt{NetworkElement})) &= G \\ &: OMSGraph \end{aligned}}$$

If $elem_1, \ldots, elem_n$ are all `NetworkElements`,

$$sem(\Gamma, Set\{elem_1, \ldots, elem_n\}) = G$$

where
$G_1 = sem(\Gamma, G_\emptyset, elem_1)$, where $G_\emptyset$ is the empty graph,
$G_2 = sem(\Gamma, G_1, elem_2)$
...
$G_n = sem(\Gamma, G_{n-1}, elem_n)$,
$G = addImports(\Gamma, G_n, [elem_1, \ldots, elem_n])$.

10.3.2.4 Semantics of network elements

$$\boxed{\begin{aligned} sem(\Gamma, G, \mathtt{NetworkElement}) &= G' \\ &: OMSGraph \end{aligned}}$$

If $networkElement$ is a `NetworkElement`,

$$sem(\Gamma, G, networkElement) = insert(G, \Gamma, networkElement.\mathtt{elementRef.iri}, networkElement.\mathtt{id})$$

10.3.2.5 Semantics of sets of excluded elements

$$\boxed{\begin{aligned} sem(\Gamma, G, Set(\mathtt{ExcludedElement})) &= G' \\ &: OMSGraph \end{aligned}}$$

10.3.3.5 Semantics of ExtendingOMS

$$\boxed{\begin{aligned} sem(\Gamma, \texttt{ExtendingOMS}) &= (\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta)) \\ &: (\mathit{LogicalEnvironment}, (\mathit{Institution}, \mathit{Signature}, \mathit{ModelClass}, \mathit{Sentences})) \end{aligned}}$$

The semantics for `ClosableOMS` has been defined above.

If O is a `RelativeClosureOMS`, $O.\texttt{closureType} = \texttt{minimize}$ and $O' = O.\texttt{closableOMS}$, then

$$sem(\Gamma, O) = (\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta))$$

where

— $\mathcal{I} = \mathbf{Inst}(O')$

— $\Sigma = \mathsf{Sig}(O')$

— $\mathcal{M} = \{M \in \mathsf{Mod}(O') \mid M \text{ is minimal in } \mathsf{Mod}(O')\}$ and "minimal" is interpreted in the pre-order defined by $M_1 \leq M_2$ if there is a ~~model homomorphism~~ homomorphism of realizations $\mathcal{M}_1 \to \mathcal{M}_2$.

— $\Delta = \perp$

— Γ' is obtained from $\Gamma'' = Env(\Gamma, O')$ by adding to $\Gamma''.imports$ a new node labeled with $(Name(O), \mathbf{Inst}(O), \mathsf{Sig}(O), \mathsf{Mod}(O), \mathsf{Th}(O))$ and an edge from the node of O' to the node of O labeled with the identity morphism on $\mathsf{Sig}(O')$.

The semantics of O is defined similarly for the other three alternatives of $O.\texttt{closureType}$, only the ~~model class~~ class of realizations differs:

— if $O.\texttt{closureType} = \texttt{maximize}$, $\mathcal{M} = \{M \in \mathsf{Mod}(O') \mid M \text{ is maximal in } \mathsf{Mod}(O')\}$

— if $O.\texttt{closureType} = \texttt{free}$, $\mathcal{M} = \{M \in \mathsf{Mod}(O') \mid M \text{ is initial in } \mathsf{Mod}(O')\}$

— if $O.\texttt{closureType} = \texttt{cofree}$, $\mathcal{M} = \{M \in \mathsf{Mod}(O') \mid M \text{ is terminal in } \mathsf{Mod}(O')\}$

Here, initial and terminal ~~models~~ realizations are defined as in category theory, see P.1.1.

10.3.3.6 Semantics of ExtendingOMS in a local environment

$$\boxed{\begin{aligned} sem(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), \texttt{ExtendingOMS})) &= (\Gamma', (\mathcal{I}', \Sigma', \mathcal{M}', \Delta')) \\ &: (\mathit{LogicalEnvironment}, (\mathit{Institution}, \mathit{Signature}, \mathit{ModelClass}, \mathit{Sentences})) \end{aligned}}$$

The semantics for `ClosableOMS` has been defined above.

The semantics for minimization selects the ~~models~~ realizations that are minimal in the class of all ~~models~~ realizations with the same interpretation for the local environment (= fixed non-logical symbols, in the terminology of circumscription).

Formally, if O is a `RelativeClosureOMS`, $O.\texttt{closureType} = \texttt{minimize}$ and $O' = O.\texttt{closableOMS}$, and $sem(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), O') = (\Gamma', (\mathcal{I}', \Sigma', \mathcal{M}', \Delta'))$ then

$$sem(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), O)) = (\Gamma'', (\mathcal{I}'', \Sigma'', \mathcal{M}'', \Delta''))$$

where

— $\mathcal{I}'' = \mathcal{I}'$

— $\Sigma'' = \Sigma'$

— $\mathcal{M}'' = \{M \in \mathcal{M} \mid M \text{ is minimal in } \{M' \in \mathcal{M} \mid M'|_\Sigma = M|_\Sigma\}\}$ and "minimal" is interpreted in the pre-order defined by $M_1 \leq M_2$ if there is a ~~model homomorphism~~ homomorphism of realizations $\mathcal{M}_1 \to \mathcal{M}_2$

— $\Delta'' = \perp$

— Γ'' is obtained from $\Gamma' = \mathit{Env}(\Gamma, O.\texttt{oms})$ by extending $\Gamma'.\mathit{imports}$ with a new node for O labeled as in the items above and with a new edge from the node of $O.\texttt{oms}$ to the node of O labeled with $((\Phi, \alpha, \beta), \sigma)$.

The semantics of a `ReductionOMS` O is

$$\mathit{sem}(\Gamma, O) = (\Gamma'', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta))$$

where

— $\mathcal{I} = J$,

— $\Sigma = \Sigma'$, when $\mathit{sem}(\Gamma, \mathsf{Sig}(O.\texttt{oms}), O.\texttt{reduction}) = ((\Phi, \alpha, \beta) : \mathbf{Inst}(O.\texttt{oms}) \to J, \sigma : \Sigma' \to \Phi(\mathsf{Sig}(O.\texttt{oms})))$,

— $\mathcal{M} = \{\beta_\Sigma(M)|_\sigma \mid M \in \mathsf{Mod}(O.\texttt{oms})\}$

— $\Delta = \bot$

— Γ'' is obtained from $\Gamma' = \mathit{Env}(\Gamma, O.\texttt{oms})$ by extending $\Gamma'.\mathit{imports}$ with a new node for O labeled as in the items above and with a new edge from the node of O to the node of $O.\texttt{oms}$ labeled with $((\Phi, \alpha, \beta), \sigma)$.

The semantics of an `ExtractionOMS` O is

$$\mathit{sem}(\Gamma, O) = (\Gamma'', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta))$$

where

— $\mathcal{I} = \mathbf{Inst}(O.\texttt{oms})$

— $\Sigma = \Sigma'$,

— $\Delta = \Delta'$

— $\mathcal{M}$ is the class of Δ-~~models~~ realizations

— Γ'' is obtained from $\Gamma' = \mathit{Env}(\Gamma, O.\texttt{oms})$ by extending $\Gamma'.\mathit{imports}$ with a new node for O labeled as in the items above and with a new edge from the node of O to the node of $O.\texttt{oms}$ labeled with the inclusion of Σ' in $\mathsf{Sig}(O.\texttt{oms})$

where $\mathit{sem}(\Gamma, (\mathsf{Sig}(O.\texttt{oms}), \mathsf{Th}(O.\texttt{oms})), O.\texttt{extraction}) = (\Sigma', \Delta')$.

The semantics of an `ApproximationOMS` O is

$$\mathit{sem}(\Gamma, O) = (\Gamma'', (\mathcal{I}'', \Sigma', \mathcal{M}, \Delta))$$

where

— $\mathcal{I}'' = \mathcal{I}'$,

— $\Sigma' = \Phi(\Sigma)$

— $\Delta = \alpha^{-1}_{\mathsf{Sig}(O.\texttt{oms})}(\mathsf{Th}(O.\texttt{oms})^\bullet) \cap \mathsf{Sen}^{\mathcal{I}'}(\mathsf{Sig}(O.\texttt{oms}))$. , i.e. that part of $\mathsf{Th}(O.\texttt{oms})$ that can be expressed in the smaller signature and logic . In practice, one looks for a finite subset that still is logically equivalent to this set.

— $\mathcal{M}$ is the class of Δ-~~models~~ realizations

— Γ'' is obtained from $\Gamma' = \mathit{Env}(\Gamma, O.\texttt{oms})$ by extending $\Gamma'.\mathit{imports}$ with a new node for O labeled as in the items above and with a new edge from the node of $O.\texttt{oms}$ to the node of O labeled with $(\rho, \iota : \Phi(\Sigma) \to \mathsf{Sig}(O.\texttt{oms}))$

where $(\rho = (\Phi, \alpha, \beta) : \mathcal{I} \to \mathcal{I}', \Sigma) = \mathit{sem}(\Gamma, (\mathbf{Inst}(O.\texttt{oms}), \mathsf{Sig}(O.\texttt{oms})), O.\texttt{approximation})$.

The semantics of a `FilteringOMS` O is defined by case distinction. Let $(\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta)) = \mathit{sem}(\Gamma, O.\texttt{oms})$ and $(c, \mathcal{I}', \Sigma', \Delta') = \mathit{sem}(\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), O.\texttt{filtering})$.

If $c = \mathit{keep}$, the semantics of O is given by

$$\mathit{sem}(\Gamma, O) = (\Gamma'', (\mathcal{I}'', \Sigma'', \mathcal{M}'', \Delta''))$$

where

— $\mathcal{I}'' = \mathcal{I}'$

— Σ'' is the smallest signature with $\Sigma' \subseteq \Sigma''$ and $\Delta' \subseteq \mathsf{Sen}(\Sigma'')$. (If this smallest signature does not exist, the semantics is undefined.)
— $\Delta'' = (\Delta \cap \mathsf{Sen}(\Sigma'')) \cup \Delta'$
— $\mathsf{Mod}(O)$ is the class of all Δ-~~models.~~ realizations
— Γ'' is obtained from Γ' by extending $\Gamma'.imports$ with a new node for O labeled as in the items above and with a new edge from the node of O to the node of $O.\texttt{oms}$ labeled with the inclusion of Σ'' in Σ.

If $c = remove$, the semantics of O is

$$sem(\Gamma, O) = (\Gamma'', (\mathcal{I}'', \Sigma'', \mathcal{M}'', \Delta''))$$

where

—
$cI'' = \mathcal{I}'$
— $\Sigma'' = \Sigma \setminus \Sigma'$
— $\Delta'' = \Delta \cap \mathsf{Sen}(\Sigma'') \setminus \Delta'$
— $\mathcal{M}''$ is the class of all $\mathsf{Th}(O)$~~models.~~ realizations
— Γ'' is obtained from Γ' by extending $\Gamma'.imports$ with a new node for O labeled as in the items above and with a new edge from the node of O to the node of $O.\texttt{oms}$ labeled with the inclusion of Σ'' in Σ.

The semantics of an `UnionOMS` O is

$$sem(\Gamma, O) = (\Gamma'', (\mathcal{I}', \Sigma', \mathcal{M}', \Delta'))$$

where

—
$cI' = \mathcal{I}$ where $\mathbf{Inst}(O_1) \bigcup \mathbf{Inst}(O_2) = (\mathcal{I}, (\Phi_1, \alpha_1, \beta_1) : \mathbf{Inst}(O_1) \to \mathcal{I}, (\Phi_2, \alpha_2, \beta_2) : \mathbf{Inst}(O_2) \to \mathcal{I})$
— $\Sigma' = \Phi_1(\mathsf{Sig}(O_1)) \cup \Phi_2(\mathsf{Sig}(O_2))$
— $\mathcal{M}' = \{M \in \mathsf{Mod}^{\mathcal{I}}(\Sigma') \mid \beta_{\Sigma_i}(M|_{\Phi_i(\mathsf{Sig}(O_i))}) \in \mathsf{Mod}(O_i),\ \text{for } i = 1, 2\}$
— $\Delta' = \alpha_1(\mathsf{Th}(O_1)) \cup \alpha_2(\mathsf{Th}(O_2))$.
— Γ'' is obtained from $\Gamma' = Env(Env(\Gamma, O_1), O_2)$ by extending $\Gamma'.imports$ with a new node for O labeled as in the items above and with edges from the nodes of O_1 and O_2, respectively, to the node of O, labeled for each $i = 1, 2$ with $(\Phi_i, \alpha_i, \beta_i, \iota_i : \Phi_i(O_i) \to \Sigma')$.

where $O_1 = O.\texttt{oms}$ and $O_2 = O.\texttt{oms2}$.

If $O.\texttt{conservativityStrength}$ is present, then O must be a conservative extension of the appropriate strength of O_1.

The semantics of an `ExtensionOMS` O is

$$sem(\Gamma, O) = (\Gamma'', (\mathcal{I}', \Sigma', \mathcal{M}', \Delta'))$$

where

— $\mathcal{I}' = \mathbf{Inst}(O.\texttt{oms}) = \mathbf{Inst}(O.\texttt{extension})$ (which means that the institutions of $O.\texttt{oms}$ and $O.\texttt{extension}$ must be the same)
— $\Sigma' = \mathsf{Sig}(O.\texttt{oms}) \cup \mathsf{Sig}(sem(\Gamma, (\mathbf{Inst}(O.\texttt{oms}), \mathsf{Sig}(O.\texttt{oms}), \mathsf{Mod}(O.\texttt{oms}), \mathsf{Th}(O.\texttt{oms})), O.\texttt{extension})$
— $\mathcal{M}' = \{M \in \mathsf{Mod}(\Sigma') \mid M|_{\mathsf{Sig}(O.\texttt{oms})} \in \mathsf{Mod}(O.\texttt{oms}) \text{ and } M|_{\mathsf{Sig}(O.\texttt{extension})} \in \mathsf{Mod}(O.\texttt{extension})\}$
— $\Delta' = \mathsf{Th}(O.\texttt{oms}) \cup \mathsf{Th}(O.\texttt{extension})$
— Γ'' is $Env(Env(\Gamma, O.\texttt{oms}), O.\texttt{extension})$.

The semantics of a `QualifiedOMS` O in the context Γ is the same as the semantics of $O.\texttt{oms}$ in the context Γ' given by the semantics of $O.\texttt{qualification}$ in the context Γ. The change of context is local to $O.\texttt{oms}$, which means that if the qualification appears as a term in a larger expression, after its analysis the context will be Γ and not Γ'. Formally,

$$sem(\Gamma, O) = (\Gamma'', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta))$$

10.3.3.16 Semantics of approximations

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, (\mathcal{I}, \Sigma), \texttt{Approximation}) & = (\rho : \mathcal{I} \to \mathcal{I}', \Sigma') \\ & : (\mathit{Morphism}, \mathit{Signature}) \end{array}}$$

If a is an `Approximation`,

$$\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma), a) = (\rho, \Sigma')$$

where $\Sigma' = \mathit{sem}(\Gamma, \Sigma, a.\texttt{removalKind}, a.\texttt{interfaceSignature})$, and ρ is the default projection (institution morphism) from $\mathcal{I}$ to $\mathit{sem}(\Gamma, a.\texttt{logicRef}) = \mathcal{I}'$, when $a.\texttt{logicRef}$ is present and the identity institution morphism on $\mathcal{I}$, when $a.\texttt{logicRef}$ is missing.

10.3.3.17 Semantics of filtering

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), \texttt{Filtering}) & = (c, \mathcal{I}', \Sigma', \Delta') \\ & : ('\mathit{keep}'|'\mathit{remove}', \mathit{Institution}, \mathit{Signature}, \mathit{Sentences}) \end{array}}$$

If f is a `Filtering` such that $f.\texttt{removalKind} = \mathit{keep}$,

$$\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), f) = (\mathit{keep}, \mathcal{I}', \Sigma', \Delta')$$

where $\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), f.\texttt{basicOMSOrSymbolList}) = (\mathcal{I}', \Sigma', \mathcal{M}', \Delta')$.

If f is a `Filtering` such that $f.\texttt{removalKind} = \mathit{remove}$,

$$\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), f) = (\mathit{remove}, \mathcal{I}', \Sigma', \Delta')$$

where $\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), f.\texttt{basicOMSOrSymbolList}) = (\mathcal{I}', \Sigma', \mathcal{M}', \Delta')$.

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), \texttt{BasicOMSOrSymbolList}) & = (\mathcal{I}', \Sigma', \mathcal{M}', \Delta') \\ & : (\mathit{Institution}, \mathit{Signature}, \mathit{Sentences}) \end{array}}$$

If O is a `BasicOMS`, we have defined $\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), O) = (\mathit{Gamma}, (\mathcal{I}', \Sigma', \mathcal{M}', \Delta'))$ and the semantics of O as a `BasicOMSOrSymbolList` is $(\mathcal{I}', \Sigma', \mathcal{M}', \Delta')$.

If s is a set of symbols, $\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), s) = (\mathcal{I}, \mathit{sem}(\Gamma, \Sigma, s), \emptyset, \emptyset)$.

10.3.3.18 Semantics of extension

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), \texttt{Extension}) & = (\Gamma', (\mathcal{I}, \Sigma', \mathcal{M}', \Delta')) \\ & : (\mathit{LogicalEnvironment}, (\mathit{Institution}, \mathit{Signature}, \mathit{ModelClass}, \mathit{Sentences})) \end{array}}$$

If e is an `Extension`,

$$\mathit{sem}(\Gamma, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta), e) = (\Gamma', (\mathcal{I}, \Sigma', \mathcal{M}', \Delta'))$$

where $(\Gamma', (\mathcal{I}, \Sigma', \mathcal{M}', \Delta')) = \mathit{sem}(\Gamma, (\Sigma, \mathcal{M}), e.\texttt{extendingOMS})$.

If $e.\texttt{conservativityStrength}$ is `model-conservative` or `implied`, the semantics is only defined if each ~~model~~ realization in $\mathcal{M}$ is the Σ-reduct of some ~~model~~ realization in $\mathcal{M}'$. In case that $e.\texttt{conservativityStrength}$ is `implied`, it is furthermore required that $\Sigma = \Sigma'$. If $e.\texttt{conservativityStrength}$ is `consequenceconservative`, the semantics is only defined if for each Σ-sentence φ, $\mathcal{M}' \models \varphi$ implies $\mathcal{M} \models \varphi$. If $e.\texttt{conservativityStrength}$ is `definitional`, the semantics is only defined if each ~~model~~ realization in $\mathcal{M}$ is the Σ-reduct of a unique ~~model~~ realization in $\mathcal{M}'$.

If $e.\texttt{extensionName}$ is present, the inclusion link is labeled with this name.

10.3.3.19 Semantics of interface signatures

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \Sigma, \texttt{RemovalKind}, \texttt{InterfaceSignature}) & = \Sigma' \\ & : \mathit{Signature} \end{array}}$$

If r is a `RemovalKind` and s is an `InterfaceSignature`,

$$\mathit{sem}(\Gamma, \Sigma, r, s) = \Sigma'$$

where

$$\Sigma' = \begin{cases} \Sigma \cap \mathit{sem}(\Gamma, \Sigma, s.\texttt{symbolList}) & \text{if} \quad r = \mathit{keep} \\ \Sigma \setminus \mathit{sem}(\Gamma, \Sigma, s.\texttt{symbolList}) & \text{if} \quad r = \mathit{remove} \end{cases}$$

10.3.3.20 Semantics of OMS definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{OMSDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

An `OMSDefinition` O extends the global environment.

$$\mathit{sem}(\Gamma, O) = \Gamma''$$

where for each of the institutions $\mathcal{I}_1, \ldots, \mathcal{I}_n$ supported by $\Gamma.\mathit{lang}$, we have
$\Gamma_1 = \Gamma'_1[\mathit{postfixLogicIRI}(O.\mathrm{omsName}, \mathcal{I}_1) \mapsto (\mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1)]$
where $\mathit{sem}(\Gamma.\mathit{logic} = \mathcal{I}_1, O.\mathrm{oms}) = (\Gamma'_1, (\mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1))$,
$\Gamma_2 = \Gamma'_2[\mathit{postfixLogicIRI}(O.\mathrm{omsName}, \mathcal{I}_2) \mapsto (\mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2)]$
where $\mathit{sem}(\Gamma'_1.\mathit{logic} = \mathcal{I}_2, O.\mathrm{oms}) = (\Gamma'_2, (\mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2))$, …,
$\Gamma'' = \Gamma'_n[\mathit{postfixLogicIRI}(O.\mathrm{omsName}, \mathcal{I}_n) \mapsto (\mathcal{I}_n, \Sigma_n, \mathcal{M}_n, \Delta_n)]$
where $\mathit{sem}(\Gamma'_{n-1}.\mathit{logic} = \mathcal{I}_n, O.\mathrm{oms}) = (\Gamma'_n, (\mathcal{I}_n, \Sigma_n, \mathcal{M}_n, \Delta_n))$

The conservativity strength annotations refer to the semantics of $O.$`oms` in the current logic of Γ. Therefore, let $(\Gamma_0, (\mathcal{I}, \Sigma, \mathcal{M}, \Delta)) = \mathit{sem}(\Gamma, O.\texttt{oms})$.

If $O.$`conservativityStrength` is `model-conservative`, the semantics is only defined if $\mathcal{M} \neq \emptyset$.
If $O.$`conservativityStrength` is `consequence-conservative`, the semantics is only defined if Δ has only tautologies[28] as signature-free[29] logical consequences.
If $O.$`conservativityStrength` is `monomorphic`, the semantics is only defined if $\mathcal{M}$ consist of exactly one isomorphism class of ~~models~~realizations.
If $O.$`conservativityStrength` is `weak-definitional`, the semantics is only defined if $\mathcal{M}$ is empty or a singleton.
If $O.$`conservativityStrength` is `definitional`, the semantics is only defined if $\mathcal{M}$ is a singleton.

10.3.3.21 Semantics of OMS references

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{OMSReference}) & = (\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta)) \\ & : (\mathit{LogicalEnvironment}, (\mathit{Institution}, \mathit{Signature}, \mathit{ModelClass}, \mathit{Sentences})) \end{array}}$$

The rule for `OMSReferences` has been given above, as `OMSReferences` are a particular case of `ClosableOMS`.

[28] A tautology is a sentence holding in every model.

[29] A signature-free sentence is one over the empty signature.

10.3.4 Semantics of OMS Mappings

10.3.4.1 Semantics of mapping definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{MappingDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

See equations for `InterpretationDefinition`, `EntailmentDefinition`, `EquivalenceDefinition`, `ConservativeExtensionDefinition` and `AlignmentDefinition`.

10.3.4.2 Semantics of interpretation definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{InterpretationDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

If d is an `InterpretationDefinition`,

$$\mathit{sem}(\Gamma, d) = \Gamma'$$

where $\Gamma' = \Gamma[d.\texttt{interpretationName} \to (G, (\rho, \sigma), L_1, L_2)]$
and G is the graph $L_1 \xrightarrow{(\rho,\sigma)} L_2$ where

- $(L_1, L_2) = \mathit{sem}(\Gamma, d.\texttt{interpretationType})$
- $\rho = (\Phi, \alpha, \beta) : \textbf{Inst}(L_1) \to \textbf{Inst}(L_2)$ is the comorphism given by $\mathit{sem}(\Gamma, d.\texttt{omsLanguageTranslation})$. If $d.\texttt{OMSLanguageTranslation}$ is missing, the default translations between the logics is selected.
- $\sigma = \mathit{sem}(\Gamma.\{\mathit{current} = (\mathit{lang}, \mathit{logic}', \mathit{ser})\}, \Phi(\mathsf{Sig}(L_1)), \mathsf{Sig}(L_2), d.\texttt{symbolMap})$, where $\Gamma.\mathit{current} = (\mathit{lang}, \mathit{logic}, \mathit{ser})$ and logic' is the target logic of ρ, or, if $d.\texttt{symbolMap}$ is missing, σ is the identity signature morphism on $\Phi(\mathsf{Sig}(L_1))$ which must be equal with $\mathsf{Sig}(L_2)$.

The semantics is only defined if $\beta_{\mathsf{Sig}(L_1)}(M_2|_\sigma) \in \mathsf{Mod}(L_1)$ for each $M_2 \in \mathsf{Mod}(L_2)$.
If the optional argument $d.\texttt{conservativityStrength}$ is

- `model-conservative`, for each ~~model~~ realization $M_1 \in |\mathsf{Mod}(L_1)|$ there must exist a ~~model~~ realization $M_2 \in |\mathsf{Mod}(L_2)|$ such that $\beta_{\mathsf{Sig}(L_1)}(M_2|_\sigma) = M_1$.
- `consequence-conservative`, for each $\mathsf{Sig}(L_1)$-sentence φ, if $\mathcal{M}_2 \models \sigma(\alpha_{\mathsf{Sig}(L_1)}(\varphi))$ then $\mathcal{M}_1 \models \varphi$.
- `not-model-conservative`, there must exist a ~~model~~ realization $M_1 \in |\mathsf{Mod}(L_1)|$ such that there is no ~~model~~ realization $M_2 \in |\mathsf{Mod}(L_2)|$ such that $\beta_{\mathsf{Sig}(L_1)}(M_2|_\sigma) = M_1$.
- `not-consequence-conservative`, there is a $\mathsf{Sig}(L_1)$-sentence φ, such that $\mathcal{M}_2 \models \sigma(\alpha_{\mathsf{Sig}(L_1)}(\varphi))$ and $\mathcal{M}_1 \not\models \varphi$.

10.3.4.3 Semantics of refinement definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{RefinementDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

If d is a `RefinementDefinition`,

$$\mathit{sem}(\Gamma, d) = \Gamma'$$

where $\Gamma' = \Gamma[d.\texttt{interpretationName} \mapsto \mathit{sem}(\Gamma, d.\texttt{refinement})]$.

10.3.4.4 Semantics of interpretation types

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{InterpretationType}) & = ((N_1, \mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1), (N_2, \mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2)) \\ & : (NodeLabel, NodeLabel) \end{array}}$$

If t is an `InterpretationType`,

$$sem(\Gamma, t) = (L_1, L_2)$$

where

— $\mathbf{Name}(L_1) = \mathbf{Name}(t.\texttt{source})$ and $\mathbf{Name}(L_2) = \mathbf{Name}(t.\texttt{target})$,

— $(\mathbf{Inst}(L_1), \mathsf{Sig}(L_1), \mathsf{Mod}(L_1), \mathsf{Th}(L_1)) = sem(\Gamma, t.\texttt{source})$,

— $(\mathbf{Inst}(L_2), \mathsf{Sig}(L_2), \mathsf{Mod}(L_2), \mathsf{Th}(L_2)) = sem(\Gamma, t.\texttt{target})$,

10.3.4.5 Semantics of refinements

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{Refinement}) & = (((G_1, G_2), \sigma, \mathcal{M}) \\ & : (OMSGraph, OMSGraph, GraphMorphism, ModelClass) \end{array}}$$

The signature of a refinement is a pair consisting of the graph of the OMS or network of OMS being refined and the graph of the OMS or network of OMS after refinement. Together with this pair the mapping is stored along which the refinement is done. Given two networks G_1 and G_2, a *network morphism* $\sigma : G_1 \to G_2$ is

1) a graph homomorphism $\sigma^G : Shape(G_1) \to Shape(G_2)$, where given a network G, its shape $Shape(G)$ is a graph with same nodes and edges as G but with no labels of nodes,
2) a natural transformation $\sigma^M : G_1 \to \sigma^G; G_2$

such that

1) for each node N_1 in G_1 labeled with $(\mathcal{I}_1, \Sigma_1, \mathcal{M}_1)$ such that $\sigma^G(N_1)$ is a node N_2 labeled with $(\mathcal{I}_2, \Sigma_2, \mathcal{M}_2)$ in G_2, there is a signature morphism $(\rho^M_{N_1}, \sigma^M_{N_1}) : (\mathcal{I}_1, \Sigma_1) \to (\mathcal{I}_2, \Sigma_2)$, where
2) $\rho^M_{N_1} = (\Phi, \alpha, \beta) : \mathcal{I}_1 \to \mathcal{I}_2$ is an institution comorphism between the logics of the two nodes and $\sigma^M_{N_1} : \Phi(\Sigma_1) \to \Sigma_2$ is a signature morphism, such that $\beta_{\Sigma_1}(M_2|_{\sigma^M_{N_1}}) \in \mathcal{M}_1$ for each $M_2 \in \mathcal{M}_2$.

A refinement ~~model~~ realization is a class $\mathcal{M}$ of pairs of families of ~~models~~ realizations compatible with the two networks. Given a network morphism $\sigma : G_1 \to G_2$ and a G_2~~model~~ -realization F, $F|_\sigma$ is defined as the family of ~~models~~ realizations $\{M_i\}_{i \in Nodes(G_1)}$ such that $M_i = F_{\sigma^G(i)}|_{\sigma^M_i}$ for each $i \in Nodes(G_1)$.

Thus, the semantics of a `Refinement` consists of

— a refinement signature (G_1, G_2),

— a network morphism σ and

— a refinement ~~model~~ realization $\mathcal{M}$.

If r is `RefinementOMS`,

$$sem(\Gamma, \texttt{r}) = ((G, G), \sigma, \mathcal{M})$$

where

— G is a graph with just one isolated node N such that $\mathbf{Name}(N) = \mathbf{Name}(r.\texttt{oms})$ and the other elements of the tuple labeling N are given by $sem(\Gamma, r.\texttt{oms})$,

— σ is the identity morphism on $\mathsf{Sig}(r.\texttt{oms})$,

— $\mathcal{M} = \{((M),(M)) \mid M \in \mathsf{Mod}(r.\texttt{oms})\}$, where (M) is the singleton family consisting of M.

If r is `RefinementNetwork`,

$$sem(\Gamma, r) = ((G,G), \sigma, \mathcal{M})$$

where $sem(\Gamma, r.\texttt{network}) = G$, σ is the identity network morphism on G and $\mathcal{M} = \{(F,F) \mid F \in \mathsf{Mod}(G)\}$.

If r is `SimpleOMSRefinement`,

$$sem(\Gamma, r) = ((G_1, G_2'), \sigma', \mathcal{M})$$

where
$sem(\Gamma, r.\texttt{refinement}) = ((G_1, G_1'), \sigma_1, \mathcal{M}_1)$,
$sem(\Gamma, r.\texttt{refinement2}) = ((G_2, G_2'), \sigma_2, \mathcal{M}_2)$,
G_1' and G_2 are both graphs with one isolated node, labeled $(name1, \mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1)$ and respectively $(name2, \mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2)$,
$sem(\Gamma, \Sigma_1, \Sigma_2, r.\texttt{omsRefinementMap}) = (\rho = (\Phi, \alpha, \beta) : \mathcal{I}_1 \to \mathcal{I}_2, \sigma : \Phi(\Sigma_1) \to \Sigma_2)$,
σ' maps the node n of G_1 to the node of G_3 and $(\sigma')_n^M = (\sigma_1)_n^M; (\rho;\sigma); (\sigma_2)^M_{\sigma_1^G(n)}$,
and ~~$\mathcal{M} = \{(M_1, M_3) \mid \exists M_2 such that (M_2, M_3) \in \mathcal{M}_2 and (M_1, M_2|_{(\rho,\sigma)}) \in \mathcal{M}_1\}$~~
$\mathcal{M} = \{(M_1, M_3) \mid \exists M_2 \text{ such that } (M_2, M_3) \in \mathcal{M}_2 \text{ and } (M_1, M_2|_{(\rho,\sigma)}) \in \mathcal{M}_1\}$.
The refinement is correct only if for each $(M,N) \in \mathcal{M}_2$, there exists $(M_1, M_2|_{(\rho,\sigma)}) \in \mathcal{M}_1$.

If r is `SimpleNetworkRefinement`,

$$sem(\Gamma, r) = ((G_1, G_2'), \sigma', \mathcal{M}')$$

where
$sem(\Gamma, r.\texttt{refinement}) = ((G_1, G_1'), \sigma_1, \mathcal{M}_1)$,
$sem(\Gamma, r.\texttt{refinement2}) = ((G_2, G_2'), \sigma_2, \mathcal{M}_2)$,
$sem(\Gamma, G_1', G_2, r.\texttt{networkRefinementMap}) = \sigma : G_1' \to G_2$,
$\sigma' = \sigma_1; \sigma; \sigma_2$
and $\mathcal{M} = \{(F_1, F_3) \mid \exists F_2 \text{ such that } (F_1, F_2|_\sigma) \in \mathcal{M}_1 \text{ and } (F_2, F_3) \in \mathcal{M}_2\}$.
The refinement is correct only if for each $(F_2, F_3) \in \mathcal{M}_2$, there is a $(F_1, F_2|_\sigma) \in \mathcal{M}_1$.

10.3.4.6 Semantics of a set of refinements

$$\boxed{\begin{aligned} sem(\Gamma, G_1, G_2, Set(\texttt{Refinement})) &= \sigma \\ &: GraphMorphism \end{aligned}}$$

If $r_1, \ldots, r_n$ are all `Refinements`,

$$sem(\Gamma, G_1, G_2, Set(r_1, \ldots, r_n)) = \sigma$$

where
$sem(\Gamma, r_1) = ((G_1^1, G_2^1), \sigma_1, \mathcal{M}_1)$, ...,
$sem(\Gamma, r_n) = ((G_1^n, G_2^n), \sigma_n, \mathcal{M}_n)$
such that $G_1 = \bigcup_{i=1,\ldots,n} G_1^i$ and no node of G_1 appears in two graphs G_1^i and G_1^j for some $i \neq j \in 1, \ldots, n$,
$G_2 = \bigcup_{i=1,\ldots,n} G_2^i$ and no node of G_2 appears in two graphs G_2^i and G_2^j for some $i \neq j \in 1, \ldots, n$,
and $\sigma : G_1 \to G_2$ is defined by $\sigma^G(n) = \sigma_i^G(n)$ if the node n comes from G_1^i and similarly for such a node n of G coming from G_1^i, we have that $\sigma_n^M = \sigma_i^M(n)$. Moreover, σ must be total on the nodes of G_1.

10.3.4.7 Semantics of refinement maps

$$\boxed{\begin{aligned} sem(\Gamma, G_1, G_2, \texttt{RefinementMap}) &= \sigma \\ &: GraphMorphism \end{aligned}}$$

If m is an `OMSRefinementMap`,

$$sem(\Gamma, G_1, G_2, m) = (name_1, name_2, \rho, \sigma)$$

where
G_1 must be a graph with just one isolated node labeled $(name_1, \mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1)$

G_2 must be a graph with just one isolated node labeled $(name_2, \mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2)$,
$sem(\Gamma, m.\mathtt{translation}) = \rho = (\Phi, \alpha, \beta) : \mathcal{I}_1 \to \mathcal{I}_2$, or if $m.\mathtt{translation}$ is missing, the default comorphism between $\mathcal{I}_1$ and $\mathcal{I}_2$,
$sem(\Gamma', \Phi(\Sigma_1), \Sigma_2, m.\mathtt{symbolMap}) = \sigma : \Phi(\Sigma_1) \to \Sigma_2$ where $\Gamma.current = (lang, logic, ser)$, $logic'$ is the target logic of (Φ, α, β), $lang'$ and ser' are the default language and serialization for $logic'$ and $\Gamma' = \Gamma.current = (lang', logic', ser')$, or, when $m.\mathtt{symbolMap}$ is missing, $\Phi(\Sigma_1)$ and Σ_2 must be the same and σ is the identity signature morphism on Σ_2.

If m is a `NetworkRefinementMap`,
$$sem(\Gamma, G_1, G_2, m) = \sigma$$
where $sem(\Gamma, G_1, G_2, m.refinements) = \sigma$.

10.3.4.8 Semantics of entailment definitions

$$\boxed{\begin{aligned} sem(\Gamma, \mathtt{EntailmentDefinition}) &= \Gamma' \\ &: LogicalEnvironment \end{aligned}}$$

If e is an `EntailmentDefinition`,
$$sem(\Gamma, e) = \Gamma'$$
where $\Gamma' = \Gamma[e.\mathtt{entailmentName} \mapsto sem(\Gamma, e.\mathtt{entailmentType})]$.

10.3.4.9 Semantics of entailment types

$$\boxed{\begin{aligned} sem(\Gamma, \mathtt{EntailmentType}) &= G \\ &: OMSGraph \end{aligned}}$$

If t is an `OMSOMSEntailment`,
$$sem(\Gamma, t) = L_2 \overset{\text{id}}{\to} L_1$$
where $\mathbf{Name}(L_1) = \mathbf{Name}(t.\mathtt{premise})$, $\mathbf{Name}(L_2) = \mathbf{Name}(t.\mathtt{conclusion})$,
$(\mathbf{Inst}(L_1), \mathsf{Sig}(L_1), \mathsf{Mod}(L_1), \mathsf{Th}(L_1)) = sem(\Gamma, t.\mathtt{premise})$, $(\mathbf{Inst}(L_2), \mathsf{Sig}(L_2), \mathsf{Mod}(L_2), \mathsf{Th}(L_2)) = sem(\Gamma, t.\mathtt{conclusion})$ such that $\mathsf{Sig}(L_1) = \mathsf{Sig}(L_2)$ and $\mathsf{Mod}(L_1) \subseteq \mathsf{Mod}(L_2)$ and id is the identity morphism on $\mathsf{Sig}(L_1)$.

If t is a `NetworkOMSEntailment`, $sem(\Gamma, t) = G$

where $sem(\Gamma, t.\mathtt{network}) = G'$ such that G' contains a node n labeled with $\mathbf{Name}(t.\mathtt{premise})$,
$sem(\Gamma, t.\mathtt{oms}) = (\mathcal{I}, \Sigma, \mathcal{M}_2, \Delta_2)$ and
$\{\mathcal{M}_n \mid \mathcal{M}$ is compatible with $G'\} \subseteq \mathcal{M}_2$. Then G extends G' with a new node whose label has the name $\mathbf{Name}(t.\mathtt{oms})$ and the other components given by $sem(\Gamma, t.\mathtt{oms})$ and with a new theorem link from this new node to the node $\mathbf{Name}(t.\mathtt{omsName})$, labeled with the identity morphism on Σ.

If t is a `NetworkNetworkEntailment`,
$$sem(\Gamma, t) = G$$
where $sem(\Gamma, t.\mathtt{premise}) = G_1$, $sem(\Gamma, t.\mathtt{conclusion}) = G_2$, such that $Shape(G_1) = Shape(G_2)$ and, for each node $i \in |Shape(G_1)|$, its names in the networks G_1 and G_2 are the same, its signatures are the same and the class of ~~models~~ realizations obtained by projecting each family of ~~models~~ realizations compatible with G_1 to the component i is included in the class of ~~models~~ realizations obtained by projecting each family of ~~models~~ realizations compatible with G_2 to the component i. Then G extends the union of G_1 and G_2 for each pair of nodes (i_1, i_2), where i_1 and i_2 identify the occurrences of the same node i in G_1 and G_2 respectively, with a theorem link from i_1 to i_2 labeled with the identity on $\mathsf{Sig}(i_1)$.

10.3.4.10 Semantics of equivalence definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{EquivalenceDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

If d is an `EquivalenceDefinition`,

$$\mathit{sem}(\Gamma, d) = \Gamma'$$

where $\Gamma' = \Gamma[d.\texttt{equivalenceName} \mapsto \mathit{sem}(\Gamma, d.\texttt{equivalenceType})]$.

10.3.4.11 Semantics of OMS equivalences

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{OMSEquivalence}) & = (G, N_1, N_2) \\ & : (\mathit{OMSGraph}, \mathit{Node}, \mathit{Node}) \end{array}}$$

If t is an `OMSEquivalence`,

$$\mathit{sem}(\Gamma, t) = (G, N_1, N_2)$$

where $O_1 = t.\texttt{oms}$, $O_2 = t.\texttt{oms2}$, $O_3 = t.\texttt{mediatingOMS}$,
$\mathit{sem}(\Gamma, (\mathcal{I}, \mathsf{Sig}(O_1) \cup \mathsf{Sig}(O_2), \mathsf{Mod}^{\mathcal{I}}(\mathsf{Sig}(O_1) \cup \mathsf{Sig}(O_2)), \emptyset), O_3) = (\Gamma', (\mathcal{I}, \Sigma, \mathcal{M}, \Delta))$
G is the graph $N_1 \xrightarrow{\iota_1} N_3 \xleftarrow{\iota_2} N_3$ where

1) N_1 is labeled with $(\mathbf{Name}(O_1), \mathbf{Inst}(O_1), \mathsf{Sig}(O_1), \mathsf{Mod}(O_1), \mathsf{Th}(O_1))$,
2) N_2 is labeled with $(\mathbf{Name}(O_2), \mathbf{Inst}(O_2), \mathsf{Sig}(O_2), \mathsf{Mod}(O_2), \mathsf{Th}(O_2))$ and
3) N_3 is labeled with $(\mathbf{Name}(O_3), \mathcal{I}, \Sigma, \mathcal{M}, \Delta)$

such that

1) $\iota_i : \mathsf{Sig}(O_i) \to \Sigma$ are signature inclusions,
2) $\mathcal{I} = \mathbf{Inst}(O_1) = \mathbf{Inst}(O_2)$ and
3) for each $i = 1, 2$ and each ~~model~~ realization $M_i \in \mathsf{Mod}(O_i)$ there exists a unique ~~model~~ realization $M \in \mathcal{M}$ such that $M|_{\mathsf{Sig}(O_i)} = M_i$.

10.3.4.12 Semantics of network equivalences

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{NetworkEquivalence}) & = (G_1, G_2, G_3) \\ & : (\mathit{OMSGraph}, \mathit{OMSGraph}, \mathit{OMSGraph}) \end{array}}$$

If t is a `NetworkEquivalence`,

$$\mathit{sem}(\Gamma, t) = (G_1, G_2, G_3)$$

where $n_1 = t.\texttt{network}$, $n_2 = t.\texttt{network2}$, $n_3 = t.\texttt{mediatingNetwork}$, $\mathit{sem}(\Gamma, n_1) = G_1$, $\mathit{sem}(\Gamma, n_2) = G_2$, $\mathit{sem}(\Gamma, n_3) = G_3$ such that G_1 and G_2 are subgraphs of G_3 and for each $i = 1, 2$ and each family of ~~models~~ realizations $\mathcal{M}_i$ compatible with G_i there is a unique family of ~~models~~ realizations $\mathcal{M}$ compatible with G_3 such that the projection of $\mathcal{M}$ to the nodes in G_i is $\mathcal{M}_i$.

10.3.4.13 Semantics of conservative extension definitions

$$\boxed{\begin{array}{ll} \mathit{sem}(\Gamma, \texttt{ConservativeExtensionDefinition}) & = \Gamma' \\ & : \mathit{LogicalEnvironment} \end{array}}$$

If d is a `ConservativeExtensionDefinition`,

$$sem(\Gamma, d) = \Gamma'$$

where $O_1 = d.\texttt{moduleType.module}$, $O_2 = d.\texttt{moduleType.whole}$, $c = d.\texttt{conservativityType}$, $\Sigma = sem(\Gamma, d.\texttt{interfaceSignature})$, $\Gamma' = \Gamma[d.\texttt{moduleName} \mapsto (G, \iota, N_2, N_1)]$ and G is the graph $N_1 \xrightarrow{\iota} N_2$ where N_1 is labeled with $(O_1, \mathbf{Inst}(O_1), \mathsf{Sig}(O_1), \mathsf{Mod}(O_1), \mathsf{Th}(O_1))$, N_2 with $(O_2, \mathbf{Inst}(O_2), \mathsf{Sig}(O_2), \mathsf{Mod}(O_2), \mathsf{Th}(O_2))$, and ι is an inclusion, when $\Sigma \subseteq \mathsf{Sig}(O_2) \subseteq \mathsf{Sig}(O_1)$ and if $c = \texttt{model-conservative}$ and for each $M \in \mathsf{Mod}(O_2)$ there is a ~~model~~ realization $M' \in \mathsf{Mod}(O_1)$ such that $M'|_\Sigma = M|_\Sigma$, or if $c = \texttt{consequence-conservative}$ and for each $\varphi \in \mathsf{Sen}(\Sigma)$, $O_1 \models \varphi$ implies $O_2 \models \varphi$.

10.3.4.14 Semantics of alignment definitions

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{AlignmentDefinition}) & = \Gamma' \\ & : LogicalEnvironment \end{array}}$$

If d is an `AlignmentDefinition`,

$$sem(\Gamma, d) = \Gamma'$$

where $sem(\Gamma, d.\texttt{alignmentType}) = (\Gamma_0, L_1, L_2)$ and $\Gamma' = \Gamma_0[d.\texttt{AlignmentName} \mapsto (G, L_1', L_2')]$,
where $(L_1', L_2') = sem(\Gamma, L_1, L_2, d.\texttt{alignmentSemantics})$,
$card = d.\texttt{alignmentCardinality}$ or, when this is missing, $card = ('1', '1')$,
$aSem = d.\texttt{alignmentSemantics}$ or, when this is missing, $aSem = \texttt{single-domain}$,
and $G = sem(\Gamma_0, L_1', L_2', card, aSem, d.\texttt{correspondence})$.

10.3.4.15 Semantics of alignment types

$$\boxed{\begin{array}{ll} sem(\Gamma, \texttt{AlignmentType}) & = (\Gamma', L_1, L_2) \\ & : (LogicalEnvironment, NodeLabel, NodeLabel) \end{array}}$$

If t is an `AlignmentType`

$$sem(\Gamma, t) = (\Gamma'', L_1, L_2)$$

where $sem(\Gamma, t.\texttt{source}) = (\Gamma', (\mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1))$, $sem(\Gamma', t.\texttt{target}) = (\Gamma'', (\mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2))$, L_1 and L_2 are the labels of the nodes of $t.\texttt{source}$ and $t.\texttt{target}$ in $\Gamma'.imports$.

10.3.4.16 Semantics of alignments

$$\boxed{\begin{array}{ll} sem(\Gamma, L_1, L_2, (\texttt{AlignmentCardinality}, \texttt{AlignmentCardinality}), \texttt{AlignmentSemantics}, Set(\texttt{Correspondence})) & = G \\ & : OMSGraph \end{array}}$$

If $card_1, card_2$ are `AlignmentCardinality`, $aSem$ is an `AlignmentSemantics` and $C = Set\{c_1, \ldots, c_n\}$ a set of `Correspondences`,

$$sem(\Gamma, L_1, L_2, (card_1, card_2), aSem, C) = G$$

where $sem(\Gamma, \mathsf{Sig}(L_1), \mathsf{Sig}(L_2), aSem, C) = (\Sigma_s, \Sigma_t, (\Sigma, \Delta), \phi_s : \Sigma_s \to \Sigma, \phi_t : \Sigma_t \to \Sigma, smap, cvalues)$,
where the semantics of the alignment is not defined in the following cases:

- if $cvalues = True$ and then at least one of the correspondences in C has a confidence value different than 1 or
- if the alignment does not have the specified cardinality, i.e.
  - if $card_1 = '?'$, then $smap$ must be injective,
  - if $card_1 = '+'$, then $smap$ must be total on the symbols of $\mathsf{Sig}(L_1)$,

$$cvalues = \begin{cases} False & conf = 1 \\ True & \text{otherwise} \end{cases}$$
$sem(\Gamma, \Sigma_1, c.generalizedTerm) = sym1$, $sem(\Gamma, \Sigma_2, c.symbolRef) = sym2$, $clist = Sequence((rel, sym1, sym2))$,

If c is a `CorrespondenceBlock`,

$$sem(\Gamma, \Sigma_1, \Sigma_2, (defaultConf, defaultRel), c) = (clist, cvalues)$$

where if c.`relation` is missing, $rel = defaultRel$, else $rel = c.$`relation`
if c.`confidence` is missing, $conf = defaultConf$, else $conf = c.$`confidence`
for all correspondences $c_1, \ldots, c_n$ in c.`correspondence`, $(clist_i, cvalues_i) = sem(\Gamma, \Sigma_1, \Sigma_2, (conf, rel), c_i)$,
$clist = clist_1 ++ \ldots ++ clist_n$ and $cvalues = \vee_{i=1,\ldots,n} cvalues_i$.

$$\boxed{\begin{array}{ll} sem(\Gamma, L_1, L_2, \texttt{AlignmentSemantics}) & = ((Name_1, \mathcal{I}'_1, \Sigma'_1, \mathcal{M}'_1, \Delta'_1), (Name_2, c\mathcal{I}'_2, \Sigma'_2, \mathcal{M}'_2, \Delta'_2)) \\ & : (NodeLabel, NodeLabel) \end{array}}$$

If s is an `AlignmentSemantics`, $L_1 = (aName, \mathcal{I}_1, \Sigma_1, \mathcal{M}_1, \Delta_1)$ and $L_2 = (aName', \mathcal{I}_2, \Sigma_2, \mathcal{M}_2, \Delta_2)$

$$sem(\Gamma, L_1, L_2, s) = (rel(L_1), rel(L_2))$$

where

$$rel(L) = \begin{cases} (L_1, L_2) & \text{if } s = \texttt{single-domain} \\ (name_1, \mathcal{I}_1, \Sigma'_1, \mathcal{M}'_1, \Delta'_1), (name_2, \mathcal{I}_2, \Sigma'_2, \mathcal{M}'_2, \Delta'_2) & \text{otherwise} \end{cases}$$

$relativize_{\mathcal{I}_1}(\Sigma_1, \Delta_1) = (\Sigma'_1, \Delta'_1)$, $\mathcal{M}'_1$ is the class of Δ'_1 ~~models~~ realizations, $name_1 =' relativized' ++aName$,
$relativize_{\mathcal{I}_2}(\Sigma_2, \Delta_2) = (\Sigma'_2, \Delta'_2)$, $\mathcal{M}'_2$ is the class of Δ'_2 ~~models~~ realizations and $name_2 =' relativized' ++aName'$.

C.3.2.3 RDF Conformance of a Modified Serialization of OWL in RDF With DOL

The serialization of OWL in RDF (regardless of the concrete *RDF* serialization employed to serialize the RDF graph that represents the OWL ontology) does not satisfy requirement (2) for RDF conformance because there is an `owl:imports` property but no class representing imports. Therefore, it is not possible to represent a concrete import, of an ontology O_1 importing an ontology O_2, as an RDF resource. However, only resources can have identifiers in RDF. RDF reification would allow for turning the statement O_1 `owl:imports` O_2 into a resource and thus giving it an identifier. However, the RDF triples required for expressing this reification, including, e.g., the triple `:import_id rdf:predicate owl:imports`, would not match the head of any rule in the mapping from RDF graphs to the OWL structural specification[30]. They would thus remain left over in the RDF graph that is attempted to be parsed into an OWL ontology, and thus violate the requirement that at the end of this parsing process, the RDF graph must be empty[31].

After extending the specification of the serialization of OWL in RDF in the following way, it satisfies the RDF conformance criteria: if the input RDF graph G considered in section 3 of **NR20** contains the pattern

i `rdf:subject` s .
i `rdf:predicate owl:imports` .
i `rdf:object` o .

and thus introduces a resource i to represent that the ontology s imports the ontology o, these three triples are removed from G. From an ontology serialized in this super-language of the serialization of OWL in RDF, one can obtain semantically equivalent ontologies (with regard to the semantics of OWL) by stripping all triples whose predicate is *rdf:subject*, *rdf:predicate* or *rdf:object*, or by adding triples that declare these three properties to be *annotation properties*.

C.4 Semantic Conformance of OWL 2 With DOL

The logic $\mathcal{SROIQ}$ underlying OWL can be formalized as an institution as follows:

Definition 15 ***OWL 2 DL.*** OWL *2 DL is the description logic (DL) based fragment of the web ontology language* OWL. *First, the simple description logic $\mathcal{ALC}$ is discussed, afterward the approach is generalized to the more complex description logic $\mathcal{SROIQ}$, which is underlying* OWL *2 DL. Signatures of the description logic $\mathcal{ALC}$ consist of a set $\mathcal{A}$ of atomic concepts, a set $\mathcal{R}$ of roles and a set $\mathcal{I}$ of individual constants. Signature morphisms are tuples of functions, one for each signature component.* ~~*Models*~~ *Realizations are first-order structures $I = (\Delta^I, .^I)$ with universe Δ^I that interpret concepts as unary and roles as binary predicates (using $.^I$). $I_1 \leq I_2$ if $\Delta^{I_1} = \Delta^{I_2}$ and all concepts and roles of I_1 are subconcepts and subroles of those in I_2. Sentences are subsumption relations $C_1 \sqsubseteq C_2$ between concepts, where concepts follow the grammar*

$$C ::= \mathcal{A} \,|\, \top \,|\, \bot \,|\, C_1 \sqcup C_2 \,|\, C_1 \sqcap C_2 \,|\, \neg C \,|\, \forall R.C \,|\, \exists R.C$$

These kind of sentences are also called TBox sentences. Sentences can also be ABox sentences, which are membership assertions of individuals in concepts (written $a : C$ for $a \in \mathcal{I}$) or pairs of individuals in roles (written $R(a, b)$ for $a, b \in \mathcal{I}, R \in \mathcal{R}$). Satisfaction is the standard satisfaction of description logics.

The logic $\mathcal{SROIQ}$ [28], which is the logical core of the Web Ontology Language OWL *2 DL*[32]*, extends $\mathcal{ALC}$ with the following constructs: (i) complex role inclusions such as $R \circ S \sqsubseteq S$ as well as simple role hierarchies such as $R \sqsubseteq S$, assertions for symmetric, transitive, reflexive, asymmetric and disjoint roles (called RBox sentences, denoted by $\mathcal{SR}$), as well as the construct $\exists R.\mathsf{Self}$ (collecting the set of 'R-reflexive points'); (ii) nominals, i.e. concepts of the form $\{a\}$, where $a \in \mathcal{I}$ (denoted by $\mathcal{O}$); (iii) inverse roles (denoted by $\mathcal{I}$); qualified and unqualified number restrictions ($\mathcal{Q}$). For details on the rather complex grammatical restrictions for $\mathcal{SROIQ}$ (e.g. regular role inclusions, simple roles) compare [28].*

OWL profiles *are syntactic restrictions of* OWL *2 DL that support specific modeling and reasoning tasks, and which are accordingly based on DLs with appropriate computational properties. Specifically,* OWL *2* EL *is designed for ontologies containing large numbers of concepts or relations,* OWL *2* QL *to support query answering over large amounts of data, and* OWL *2* RL *to support scalable reasoning using rule languages (*EL*,* QL*, and* RL *for short) .*

[30] **NR20**, section 3

[31] See the last sentence of section 3.2.5 of **NR20**

[32] See also `http://www.w3.org/TR/owl2-overview/`

Definition 18 *Given a datatype map D and a vocabulary V over D, an interpretation*

$$I = (\Delta_I, \Delta_D, \cdot^C, \cdot^{OP}, \cdot^{DP}, \cdot^I, \cdot^{DT}, \cdot^{LT}, \cdot^{FA}, NAMED)$$

for D and V is a 10-tuple with the following structure:

- Δ_I *is a nonempty set called the object domain.*
- Δ_D *is a nonempty set disjoint with Δ_I called the data domain such that $(DT)^{DT} \subseteq \Delta_D$ for each datatype $DT \in V_{DT}$.*
- $\cdot^C$ *is the class interpretation function that assigns to each class $C \in V_C$ a subset $(C)^C \subseteq \Delta_I$ such that*
  - $(owl{:}Thing)^C = \Delta_I$ *and*
  - $(owl{:}Nothing)^C = \emptyset$.
- $\cdot^{OP}$ *is the object property interpretation function that assigns to each object property $OP \in V_{OP}$ a subset $(OP)^{OP} \subseteq \Delta_I \times \Delta_I$ such that*
  - $(owl{:}topObjectProperty)^{OP} = \Delta_I \times \Delta_I$ *and*
  - $(owl{:}bottomObjectProperty)^{OP} = \emptyset$.
- $\cdot^{DP}$ *is the data property interpretation function that assigns to each data property $DP \in V_{DP}$ a subset $(DP)^{DP} \subseteq \Delta_I \times \Delta_D$ such that*
  - $(owl{:}topDataProperty)^{DP} = \Delta_I \times \Delta_D$ *and*
  - $(owl{:}bottomDataProperty)^{DP} = \emptyset$.
- $\cdot^I$ *is the individual interpretation function that assigns to each individual $a \in V_I$ an element $(a)^I \in \Delta_I$.*
- $\cdot^{DT}$ *is the datatype interpretation function that assigns to each datatype $DT \in V_{DT}$ a subset $(DT)^{DT} \subseteq \Delta_D$ such that*
  - $\cdot^{DT}$ *is the same as in D for each datatype $DT \in N_{DT}$, and*
  - $(rdfs{:}Literal)^{DT} = \Delta_D$.
- $\cdot^{LT}$ *is the literal interpretation function that is defined as $(lt)^{LT} = (LV, DT)^{LS}$ for each $lt \in V_{LT}$, where LV is the lexical form of lt and DT is the datatype of lt.*
- $\cdot^{FA}$ *is the facet interpretation function that is defined as $(F, lt)^{FA} = (F, (lt)^{LT})^{FS}$ for each $(F, lt) \in V_{FA}$.*
- $NAMED$ *is a subset of Δ_I such that $(a)^I \in NAMED$ for each named individual $a \in V_I$.*

□

The institution $\mathcal{SROIQ}(D)$ underlying OWL is now defined as follows:

Definition 19 — *An $\mathcal{SROIQ}(D)$ signature is a pair (D, V), where D is a datatype map and V a vocabulary over D.*

- *Given $\mathcal{SROIQ}(D)$ signatures (D, V) and (D', V'), a $\mathcal{SROIQ}(D)$ signature morphism $\sigma\colon (D, V) \to (D', V')$ only exists if $D \subseteq D'$. In this case, such a signature morphism consists of*
  - *a map* $\sigma_C\colon V_C \to V'_C$,
  - *a map* $\sigma_{OP}\colon V_{OP} \to V'_{OP}$,
  - *a map* $\sigma_{DP}\colon V_{DP} \to V'_{DP}$,
  - *a map* $\sigma_I\colon V_I \to V'_I$,
  - *a map $\sigma_{DT}\colon V_{DT} \to V'_{DT}$ that is the identity on $N_{DT} \cup \{rdfs{:}Literal\}$,*
  - *a map* $\sigma_{LT}\colon V_{LT} \to V'_{LT}$
- *The sentences for a signature are definded as in the direct model-theoretic semantics of OWL [61]. Sentence translation is substitution of symbols.*
- *(D, V)~~-models~~ -realizations are interpretations for D and V. Morphisms of (D, V)~~-model morphisms~~ -models are maps between the domains Δ_I preserving membership in classes and properties, where Δ_D is mapped identically. ~~Model reducts~~ Reducts of realizations are built by first translating along the signature morphism and then looking up the interpretation in the ~~model~~ realization to be reduced.*

— *The satisfaction relation is defined as in direct model-theoretic semantics of OWL [61].*

□

Remark: strictly speaking, the institution defined above is *OWL 2 DL without restrictions* in the sense of [68]. The reason is that in an institution, the sentences can be used for arbitrary formation of theories. This is related to the presence of DOL's union operator on OMS. OWL 2 DL's specific restrictions on theory formation can be modeled *inside* this institution, as a constraint on OMS. This constraint is generally not preserved under unions or extensions. DOL's multi-logic capability allows the clean distinction between ordinary OWL 2 DL and OWL 2 DL without restrictions.

C.4.1 Relativization in OWL

Definition 20 *Given an* OWL *theory* $T = ((C, R, I), \Delta)$*, the* relativization *of* T*, denoted* $\tilde{T}$*, is the theory* $((C', R, I), \Delta')$ *where*

— $C' = C \cup \{\top_T\}$

— Δ' *contains axioms stating that:*

 — *each concept in* C *is subsumed by* $\top_T$*,*

 — *each individual in* I *is an instance of* $\top_T$*,*

 — *each role* r *has its domain and range intersected with* $\top_T$*, if they are present in* Δ*, otherwise they are* $\top_T$*,*

and, for each sentence $e \in \Delta$*, the sentence* $\alpha(e)$*, obtained by replacing the concepts in* e *as follows: are made:*

 — *each occurence of* $\top$ *is replaced with* $\top_T$*,*

 — *each occurence of* $\neg C$ *is replaced with* $\top_T \sqcap \neg C$*,*

 — *each occurence of* $\forall\ r \bullet C$ *is replaced with* $\top_T \sqcap \forall\ r \bullet C$*.*

Definition 21 *Given an* OWL *theory* $T = ((C, R, I), \Delta)$*, we define* $\beta : Mod^{\mathsf{OWL}}(\tilde{T}) \to Mod^{\mathsf{OWL}}(T)$ *as follows: if* $M' \in Mod^{\mathsf{OWL}}(\tilde{T})$*, then* $M = \beta(M')$ *has as universe* Δ^M *the set* $(\top_T)^{M'}$ *and each concept, role and individual are interpreted in* M *in the same way as in* M'*. Since* M' *is a* Δ'*~~-model~~-realization, we get that* M *is indeed a* (C, R, I)*~~-model~~-realization and moreover* $M \models \Delta$*.*

NOTE If $T = ((C, R, I), \Delta)$ is an OWL theory, M' is a $\tilde{T}$~~-model~~-realization and e is a (C, R, I)-sentence, we have that $M' \models \alpha(e)$ if and only if $\beta(M') \models e$.

C.4.2 Translating correspondences to a bridge theory in OWL

We define the function $theoryOfCorrespondences_{\mathsf{OWL}}$ that takes as arguments the assumption made on the semantics of the alignment where the correspondences come from, the signatures of the two ontologies being aligned and a list of processed correspondences, in the sense that the default correspondence and any correspondence block, if present, are replaced with the lists of single correspondences they induce, represented as triples of the form $(relation, sourceSymbol, targetSymbol)$. The result of the function is a co-span of theories: $(\Sigma_s, \emptyset) \xrightarrow{\varphi_s} (\Sigma, \Delta) \xleftarrow{\varphi_t} (\Sigma_t, \emptyset)$. Intuitively, Σ_s and Σ_t gather the symbols of the aligned ontologies that appear in the list of correspondences passed as an argument, while (Σ, Δ) contains an OWL sentence representing each correspondence.

We distinguish three cases.

1) Single domain:
 — no other symbols occur in the signatures Σ_s and Σ_t than the ones that appear in correspondences: $\Sigma_s = (C_s, R_s, I_s)$ and $\Sigma_t = (C_t, R_t, I_t)$, where C_s, R_s and I_s are the sets of all concept names, roles and individuals that appear in the list of correspondences as source symbols and C_t, R_t and I_t are the sets of all concept names, roles and individuals that appear in the list of correspondences as target symbols.

Annex D
(informative)
Conformance of Common Logic with DOL

D.1 Abstract Syntax Conformance of Common Logic With DOL

The metaclass `Text` is a subclass (in the sense of SMOF **NR26** multiple classification) of `NativeDocument`. The metaclass `Sentence` is a subclass (in the sense of SMOF **NR26** multiple classification) of `BasicOMS`.

D.2 Serialization Conformance of Common Logic With DOL

The semantic conformance of Common Logic (as specified in **NR7**) with DOL is established in [56].

The XCF dialect of Common Logic has a serialization that satisfies the criteria for XML conformance. The CLIF dialect of Common Logic has a serialization that satisfies the criteria for text conformance.

D.3 Semantic Conformance of Common Logic With DOL

Common Logic can be defined as an institution as follows:

Definition 22 ***Common Logic.*** *A common logic signature Σ (called vocabulary in Common Logic terminology) consists of a set of names, with a subset called the set of discourse names, and a set of sequence markers. An signature morphism maps names and sequence markers separately, subject to the requirement that a name is a discourse name in the smaller signature if and only if it is one in the larger signature. A Σ~~-model~~ -realization $I = (UR, UD, rel, fun, int, seq)$ consists of a set UR, the universe of reference, with a non-empty subset $UD \subseteq UR$, the universe of discourse, and four mappings:*

- *rel from UR to subsets of $UD^* = \{\langle x_1, \ldots, x_n \rangle \mid x_1, \ldots, x_n \in UD\}$ (i.e., the set of finite sequences of elements of UD);*
- *fun from UR to total functions from UD^* into UD;*
- *int from names in Σ to UR, such that $int(v)$ is in UD if and only if v is a discourse name;*
- *seq from sequence markers in Σ to UD^*.*

A Σ-sentence is a first-order sentence, where predications and function applications are written in a higher-order like syntax: $t(s)$. Here, t is an arbitrary term, and s is a sequence term, which can be a sequence of terms $t_1 \ldots t_n$, or a sequence marker. A predication $t(s)$ is interpreted by evaluating the term t, mapping it to a relation using rel, and then asking whether the sequence given by the interpretation s is in this relation. Similarly, a function application $t(s)$ is interpreted using fun. Otherwise, interpretation of terms and formulae is as in first-order logic. A further difference to first-order logic is the presence of sequence terms (namely sequence markers and juxtapositions of terms), which denote sequences in UD^*, with term juxtaposition interpreted by sequence concatenation. Note that sequences are essentially a non-first-order feature that can be expressed in second-order logic.

~~Model reducts~~ Reducts of realizations are defined in the following way: Given a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ and a Σ_2~~-model~~ -realization $I_2 = (UR, UD, rel, fun, int, seq)$, $I|_\sigma = (UR, UD, rel, fun, int \circ \sigma, seq \circ \sigma)$.

Given two CL~~models~~ *realizations $I_1 = (UR_1, UD_1, rel_1, fun_1, int_1, seq_1)$ and $I_2 = (UR_2, UD_2, rel_2, fun_2, int_2, seq_2)$, a homomorphism of realizations $h : I_1 \to I_2$ is a function $h : UR_1 \to UR_2$ such that*

- *h restricts to $k : UD_1 \to UD_2$,*
- *for each $x \in UR_1$ and $s \in UD_1^*$, if $s \in rel_1(x)$, then $k^*(s) \in rel_2(h(x))$[34],*
- *for each $x \in UR_1$, $k \circ fun_1(x) = fun_2(h(x)) \circ k^*$,*

[34] k^* is the extension of h to sequences.

Annex E
(informative)
Conformance of RDF and RDF Schema with DOL

E.1 Abstract Syntax Conformance of RDF and RDF Schema With DOL

The metaclass `Document` is a subclass (in the sense of SMOF **NR26** multiple classification) of `NativeDocument`. The metaclass `Triple` is a subclass (in the sense of SMOF **NR26** multiple classification) of `BasicOMS`.

E.2 Serialization Conformance of RDF and RDF Schema With DOL

The way of representing RDF Schema ontologies as RDF graphs satisfies the criteria for RDF conformance.

E.3 Semantic Conformance of RDF and RDF Schema With DOL

The semantic conformance of RDF Schema (as specified in **NR18**) with DOL is established in [56].

Definition 23 (RDF and RDF Schema) *The institutions for the Resource Description Framework (*RDF*) and* RDF *Schema (also known as* RDFS*), respectively, are defined in the following [44]. Both* RDF *and* RDFS *are based on a logic called* bare RDF *(*SimpleRDF*), which consists of triples only (without any predefined resources).*

A signature $\mathbf{R_s}$ *in* SimpleRDF *is a set of resource references. For* $sub, pred, obj \in \mathbf{R_s}$*, a triple of the form* $(sub, pred, obj)$ *is a sentence in* SimpleRDF*, where sub, pred, obj represent subject name, predicate name, object name, respectively. An* $\mathbf{R_s}$ ~~*-model*~~ *-realization* $M = \langle R_m, P_m, S_m, EXT_m \rangle$ *consists of a set* R_m *of resources, a set* $P_m \subseteq R_m$ *of predicates, a mapping function* $S_m : \mathbf{R_s} \to R_m$*, and an extension function* $EXT_m : P_m \to \mathcal{P}(R_m \times R_m)$ *mapping every predicate to a set of pairs of resources. Satisfaction is defined as follows:*

$$\mathfrak{M} \models_{\mathbf{R_s}} (sub, pred, obj) \Leftrightarrow (S_m(sub), (S_m(obj)) \in EXT_m(S_m(pred)).$$

Both RDF *and* RDFS *are built on top of* SimpleRDF *by fixing a certain standard vocabulary both as part of each signature and in the* ~~*models*~~*realizations.*

Actually, the standard vocabulary is given by a certain theory. In case of RDF*, it contains e.g. resources* `rdf:type` *and* `rdf:Property` *and* `rdf:subject`*, and sentences like, e.g.*

$$(\texttt{rdf:type}, \texttt{rdf:type}, \texttt{rdf:Property}) \text{ , and}$$
$$(\texttt{rdf:subject}, \texttt{rdf:type}, \texttt{rdf:Property}) \text{ .}$$

In the ~~*models*~~*realizations, the standard vocabulary is interpreted with a fixed* ~~*model*~~*realization. Moreover, for each* RDF~~*-model*~~ *-realization* $M = \langle R_m, P_m, S_m, EXT_m \rangle$*, if* $p \in P_m$*, then it must hold* $(p, S_m(\texttt{rdf:Property})) \in EXT_m(\texttt{rdf:type})$*. For* RDFS*, similar conditions are formulated (here, for example also the subclass relation is fixed).*

In the case of RDFS*, the standard vocabulary contains more elements, like* `rdfs:domain`*,* `rdfs:range`*,* `rdfs:Resource`*,* `rdfs:Literal`*,* `rdfs:Datatype`*,* `rdfs:Class`*,* `rdfs:subClassOf`*,* `rdfs:subPropertyOf`*,* `rdfs:member`*,* `rdfs:Container`*,* `rdfs:ContainerMembershipProperty`*.*

There is also OWL *Full, an extension of* RDFS *with resources such as* `owl:Thing` *and* `owl:oneOf`*, tailored towards the representation of* OWL *[24].* □

$$\mathsf{Boolean}, \mathsf{UnlimitedNatural}, \mathsf{Integer}, \mathsf{Real}, \mathsf{String}$$

$$\text{Generalizations: } \mathsf{Point} \leq \mathsf{Unit}, \mathsf{Linear} \leq \mathsf{Unit}$$

$$\begin{gathered}\text{Properties: } \mathsf{Line.linear} : \mathsf{Set}[\mathsf{Boolean}], \mathsf{Track.linear} : \mathsf{Set}[\mathsf{Boolean}], \\ \mathsf{Net}\blacklozenge\mathsf{station} : \mathsf{Set}[\mathsf{Station}], \mathsf{Net}\blacklozenge\mathsf{line} : \mathsf{Set}[\mathsf{Line}], \\ \mathsf{Station.net} : \mathsf{Set}[\mathsf{Net}], \mathsf{Station}\blacklozenge\mathsf{unit} : \mathsf{Set}[\mathsf{Unit}], \mathsf{Station}\blacklozenge\mathsf{track} : \mathsf{Set}[\mathsf{Track}], \\ \mathsf{Line.net} : \mathsf{Set}[\mathsf{Net}], \mathsf{Line.linear} : \mathsf{Set}[\mathsf{Linear}], \\ \mathsf{Connector.unit} : \mathsf{Set}[\mathsf{Unit}], \\ \mathsf{Unit.station} : \mathsf{Set}[\mathsf{Station}], \mathsf{Unit.connector} : \mathsf{Set}[\mathsf{Connector}], \\ \mathsf{Track.station} : \mathsf{Set}[\mathsf{Station}], \mathsf{Track.linear} : \mathsf{Set}[\mathsf{Linear}], \\ \mathsf{Linear.track} : \mathsf{Set}[\mathsf{Track}], \mathsf{Linear.line} : \mathsf{Set}[\mathsf{Line}]\end{gathered}$$

$$\begin{gathered}\text{Associations: } \mathsf{L2L}(\mathsf{line} : \mathsf{Set}[\mathsf{Line}], \mathsf{linear} : \mathsf{Set}[\mathsf{Linear}]), \\ \mathsf{L2T}(\mathsf{linear} : \mathsf{Set}[\mathsf{Linear}], \mathsf{track} : \mathsf{Set}[\mathsf{Track}]), \\ \mathsf{C2U}(\mathsf{connector} : \mathsf{Set}[\mathsf{Connector}], \mathsf{unit} : \mathsf{Set}[\mathsf{Unit}]) \\ \mathsf{N2S}(\mathsf{net} : \mathsf{Set}[\mathsf{Net}], \blacklozenge\mathsf{station} : \mathsf{Set}[\mathsf{Station}]), \\ \mathsf{N2L}(\mathsf{net} : \mathsf{Set}[\mathsf{Net}], \blacklozenge\mathsf{line} : \mathsf{Set}[\mathsf{Line}]), \\ \mathsf{S2U}(\mathsf{station} : \mathsf{Set}[\mathsf{Station}], \blacklozenge\mathsf{unit} : \mathsf{Set}[\mathsf{Unit}]), \\ \mathsf{S2T}(\mathsf{station} : \mathsf{Set}[\mathsf{Station}], \blacklozenge\mathsf{track} : \mathsf{Set}[\mathsf{Track}])\end{gathered}$$

Here all member ends are owned by class/data types.

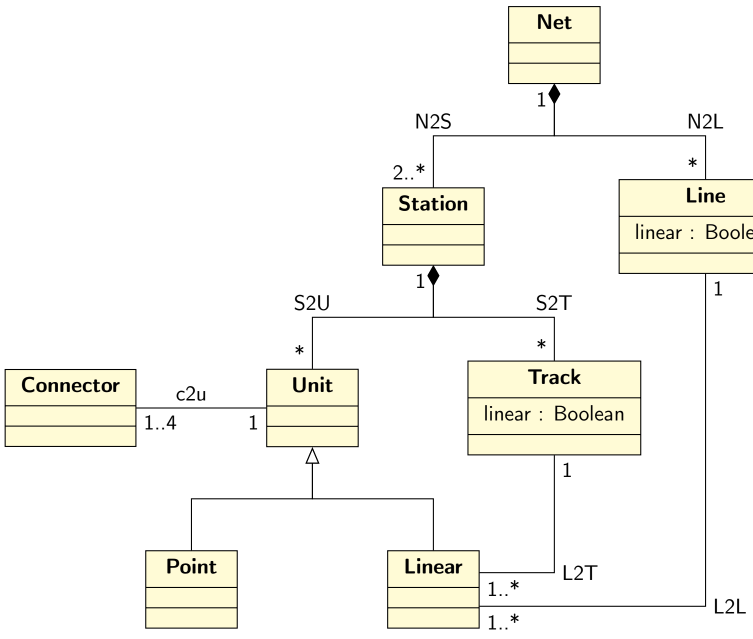

Figure F.1 – Sample UML class model

F.4.3 Realizations

As stated above, ~~models (in the sense of the term defined in clause 4)~~ realizations of UML class models are obtained via a translation to Common Logic.

For a classifier net $\Sigma = ((C, \leq_C), P, O, A)$, a Common Logic theory $\mathsf{CL}(\Sigma)$ is defined consisting of:

– for $c \in C$, a predicate[43] $\mathsf{CL}(c)$, such that

[43]Strictly speaking, this is just a name.

```
(exists (x_1 ··· x_r)
        (and (CL($c_1$) $x_1$) ··· (CL($c_r$) x_r)
             (= t (form:sequence-insert x_1 (···
                  (form:sequence-insert x_r form:empty-sequence))···))))))
```

In case that all the τ_i are omitted (or, equivalently, equal to Set), the representation is simplified to an r-ary predicate:

```
(forall (x_1 x_2 ··· x_r)
        (if (CL($a$) x_1 x_2 ··· x_r) (and (CL($c_1$) x_1) ··· (CL($c_r$) $x_r$))))
```

– the interpretation of a member end of a binary association declaration owned by a class/data type coincides with the interpretation of the attribute: if for $i \in \{1, 2\}$, $\mathbf{a}.p_i : \tau_i[c_i]$ for $\mathbf{a} = a(p_1 : \tau_1[c_1], p_2 : \tau_2[c_2]) \in A$ is owned by $c \in C$ with $c.p_i : \tau_i[c_i] \in P$, then

```
(forall (o s)
        (if (CL($c.p$) o s)
            (= s (form:seq2CL($\tau_i$) (form:select$i$ o CL($a$))))))
```

If $\mathbf{a}$ is represented in simplified form, then instead the following is used

```
(forall (o s)
        (if (CL($c.p$) o s)
            (forall (x) (iff (member x s) (CL($a$) o x)))))
```

– For the compositions, let $c^1 \blacklozenge p^1 : \tau[c'^1], \ldots, c^k \blacklozenge p^k : \tau[c'^k]$ be all the composite attributes in P and $\mathbf{a}^1 = a^1(p_1^1 : \mathsf{Set}[c_1^1], \blacklozenge p_2^1 : \tau_2^{(1)}[c_2^1]), \ldots, \mathbf{a}^l = a^{(l)}(p_1^l : \mathsf{Set}[c_1^l], \blacklozenge p_2^l : \tau_2^{(1)}[c_2^l])$ all the composite binary associations in A. Abbreviate

```
(or (CL($c^1.p^1$) o x) ··· (CL($c^k.p^k$) o x)
    (form:sequence-member (form:pair o x) CL($a^1$)) ···
      (form:sequence-member (form:pair o x) CL($a^l$))))
```

by `(owner o x)`, where, for each binary association a^j represented in the simplified way, `(CL(`a^j`) o i)` replaces `(form:sequence-member (form:pair o x) CL(`a^j`))`. Then

```
(forall (o1 o2 x)
        (if (and (owner o1 x) (owner o2 x))
            (= o1 o2)))
```

It is straightforward to extend CL from signatures to signature morphisms.

~~Models~~Realizations. A Σ-~~model~~ realization of the UML class model institution is just a CL(Σ)-~~model~~ realization in Common Logic. That is, the UML class model institution inherits ~~models~~ realizations from Common Logic. Moreover, ~~model reducts~~ reducts of realizations are inherited as well, using the action of CL on signature morphisms.

I.5.2 QL $\rightarrow$ OWL and DL-Lite$_R$ $\rightarrow$ $\mathcal{SROIQ}(D)$

QL $\rightarrow$ OWL is the sublanguage inclusion obtained by the syntactic restriction according to the definition of QL, see **NR6**. Since by definition, DL-Lite$_R$ is a syntactic restriction of $\mathcal{SROIQ}(D)$, DL-Lite$_R$ $\rightarrow \mathcal{SROIQ}(D)$ is the corresponding sublogic inclusion.

I.5.3 RL $\rightarrow$ OWL and RL $\rightarrow$ $\mathcal{SROIQ}(D)$

RL $\rightarrow$ OWL is the sublanguage inclusion obtained by the syntactic restriction according to the definition of RL, see **NR6**. Since by definition, RL is a syntactic restriction of $\mathcal{SROIQ}(D)$, RL $\rightarrow \mathcal{SROIQ}(D)$ is the corresponding sublogic inclusion.

I.5.4 SimpleRDF $\rightarrow$ RDF

SimpleRDF $\rightarrow$ RDF is an obvious inclusion, except that SimpleRDF resources need to be renamed if they happen to have a predefined meaning in RDF. The model translation needs to forget the fixed parts of RDF~~models~~realizations. Since this part can always reconstructed in a unique way, the result is an isomorphic model translation.

I.5.5 RDF $\rightarrow$ RDFS

This is entirely analogous to SimpleRDF $\rightarrow$ RDF.

I.5.6 SimpleRDF $\rightarrow$ $\mathcal{SROIQ}(D)$

A SimpleRDF signature is translated to $\mathcal{SROIQ}(D)$ by providing a class P and three roles *sub*, *pred* and *obj* (these reify the extension relation), and one individual per SimpleRDF resource. A SimpleRDF triple (s, p, o) is translated to the $\mathcal{SROIQ}(D)$ sentence

$$\top \sqsubseteq \exists U.(\exists sub.\{s\} \sqcap \exists pred.\{p\} \sqcap \exists obj.\{o\}).$$

From an $\mathcal{SROIQ}(D)$ ~~model~~ realization $\mathcal{I}$, obtain a SimpleRDF~~model~~ realization by inheriting the universe and the interpretation of individuals (then turned into resources). The interpretation $P^{\mathcal{I}}$ of P gives P_m, and EXT_m is obtained by de-reifying, i.e.

$$EXT_m(x) := \{(y, z) \mid \exists u.(u, x) \in pred^{\mathcal{I}}, (u, y) \in sub^{\mathcal{I}}, (u, z,) \in obj^{\mathcal{I}}\}.$$

RDF $\rightarrow \mathcal{SROIQ}(D)$ is defined similarly. The theory of RDF built-ins is (after translation to $\mathcal{SROIQ}(D)$) added to any signature translation. This ensures that the model translation can add the built-ins.

I.5.7 OWL $\rightarrow$ FOL

I.5.7.1 Translation of signatures

$\Phi((\mathbf{C}, \mathbf{R}, \mathbf{I})) = (F, P)$ with

— function symbols: $F = \{a^{(1)} | a \in \mathbf{I}\}$

— predicate symbols $P = \{A^{(1)} | A \in \mathbf{C}\} \cup \{R^{(2)} | R \in \mathbf{R}\}$

I.5.7.2 Translation of sentences

Concepts are translated as follows:

I.5.8 $FOL \to$ CL

This comorphism maps classical first-order logic (FOL) to Common Logic.

A FOL signature is translated to CL.Fol by turning all constants into discourse names, and all other function symbols and all predicate symbols into non-discourse names. A FOL sentence is translated to CL.Fol by a straightforward recursion, the base being translations of predications:

$$\alpha_\Sigma(P(t_1,\dots,t_n)) = (P\ \alpha_\Sigma(t_1)\ \dots\ \alpha_\Sigma(t_n))$$

Within terms, function applications are translated similarly:

$$\alpha_\Sigma(f(t_1,\dots,t_n)) = (f\ \alpha_\Sigma(t_1)\ \dots\ \alpha_\Sigma(t_n))$$

A CL.Fol ~~model~~ realization is translated to a FOL ~~model~~ realization by using the universe of discourse as FOL universe. The interpretation of constants is directly given by the interpretation of the corresponding names in CL.Fol. The interpretation of a predicate symbol P is given by using $rel^M(int^M(P))$ and restricting to the arity of P; similarly for function symbols (using fun^M). Both the satisfaction condition and model-expansiveness of the comorphism are straightforward.

I.5.9 OWL $\to$ CL

This comorphism is the composition of the comorphisms described in the previous two sections.

I.5.10 UML class models $\to$ CL

This translation has been described in annex F. Translation of signatures is detailed in section F.4.3, translation of sentences in section F.4.5. ~~Models~~ Realizations are translated identically.

I.5.11 $FOL \to$ CASL

This is an obvious sublogic.

I.5.12 UML class model to OWL

Let $\Sigma = ((C, \leq_C), P, O, A, M)$ be a *class/data type net* representing a UML class model as described in annex F. This net can be translated to OWL2 using the approach described in [76]. The ontology is extended by translating parts of this net and its multiplicity constraints $\mathit{Mult}(\Sigma)$:

— For each class $c \in C$ with superclasses $c_1, c_2, \dots, c_n \in C$ (i.e. $c \leq_C c_i$ for $i = 1,\dots,n$):

```
Class: c
        SubClassOf: c1
        ...
        SubClassOf: cn
```

— For each attribute declaration $c.p : c'$ in P

```
ObjectProperty: p
        Domain: c
        Range: c'
```

— For each attribute multiplicity $n \leq c.p : \tau[c']$ in $\mathit{Mult}(\Sigma)$ extend the description of class c by:

```
SubClassOf: p min n c'
```

Annex P
(informative)
Introduction to Category Theory

P.1 Categories

Definition 25 *A category C consists of*

— *a class of* objects*, denoted $|C|$,*
— *for each two objects a and b, a class of* morphisms *(or* arrows*), denoted $C(a,b)$,*
— *for each three objects a,b and c, a* composition *operation, denoted $;: C(a,b) \times C(b,c) \to C(a,c)$ such that the following axioms hold:*
  — *if $f \in C(a,b)$, $g \in C(b,c)$ and $h \in C(c,d)$ for four objects a,b,c,d, then $f;(g;h) = (f;g);h$*
  — *for each object a there is a morphism $id_a \in C(a,a)$ such that for every $f \in C(a,b)$ and every $g \in C(b,a)$ for some object b we have that $id_a; f = f$ and $g; id_a = g$.*

Example $\mathbb{S}et$ is the category whose class of objects is the class of all sets, $\mathbb{S}et(A,B)$ is the set of all functions from A to B for any sets A and B, id_A is the identity function on a set A and the composition is the usual composition of functions.

Example $\mathbb{R}el$ is the category whose class of objects is the class of all sets, $\mathbb{R}el(A,B)$ is the class of all relations $R \subseteq A \times B$, for any sets A and B, id_A is the diagonal relation $\{(a,a) \mid a \in A\}$ for a set A and the composition of $R \in \mathbb{R}el(A,B)$ with $S \in \mathbb{R}el(B,C)$ for three sets A, B, C is defined as $\{(a,c) \mid \text{exists } b \in B \text{ such that } (a,b) \in R \text{ and } (b,c) \in S\}$.

Example The category of unsorted first-order signatures has as objects tuples of the form $F = (F_i)_{i \in \mathbb{N}}$ where F_i is a set (of function symbols of arity i, for each natural number i). Given two objects F and G, a morphism $\sigma : F \to G$ is a family of functions $(\sigma_i : F_i \to G_i)_{i \in \mathbb{N}}$, which means that the arities of function symbols are preserved by morphisms. The identity morphism for an object F is the family of identity functions $(id_{F_i})_{i \in \mathbb{N}}$ and the composition is defined component-wise: if $\sigma : F \to G$ and $\tau : G \to H$ are signature morphisms between the signatures F, G and H, then $\sigma;\tau = (\sigma_i;\tau_i)_{i \in \mathbb{N}}$.

Example Given an unsorted first-order signature F, a ~~model~~ realization M of F consists of an universe M_U together with an interpretation of each function symbol $f \in F_i$ as a function M_f taking i arguments in M_U with result in M_U. Given two such ~~models~~ realizations M and N, a ~~model homomorphism~~ homomorphism of realizations $m : M \to N$ is a function $m : M_U \to N_U$ such that for each $i \in \mathbb{N}$ and each $f \in F_i$ we have that $m(M_f(x_1, \ldots, x_n)) = N_f(m(x_1), \ldots, m(x_n))$ for every $x_1, \ldots x_n$ in M_U. The identity function on M_U is a ~~model homomorphism~~ homomorphism of realizations on M and the composition is the usual composition of functions. This gives us the category of first-order ~~models~~ realizations of F.

Definition 26 *Let C be a category. Its dual or opposite category, denoted C^{op}*

— *has the same objects as C: $|C^{op}| = |C|$,*
— *for two objects $a, b \in |C|$, $C^{op}(a,b) = C(b,a)$,*
— *$;^{op} : C^{op}(a,b) \times C^{op}(b,c) \to C^{op}(a,c)$ is defined as $f;^{op} g = g;f$ for any $f \in C^{op}(a,b) = C(b,a)$ and $g \in C^{op}(b,c) = C(c,b)$. The result $g;f$ is a morphism in $C(c,a) = C^{op}(a,c)$,*
— *for each object a, $id_a \in C^{op}(a,a) = C(a,a)$ is the identity w.r.t. the composition $;^{op}$.*

Definition 27 *An object A is called an initial object in a category C if for each object B of C there is exactly one morphism from A to B.*

Definition 28 *An object A is called a terminal object in a category C if for each object B of C there is exactly one morphism from B to A.*

Example In $\mathbb{S}et$, the empty set is the initial object and each singleton set is a terminal object.

P.1.1 Limits and colimits

Definition 29 *A* network[51] *in a category C is a functor $D : G \to C$, where G is a small category*[52]*, and can be thought of as the shape of the graph of interconnections between the objects of C selected by the functor D.*

Definition 30 *A* cocone *of a network $D : G \to C$ consists of an object c of C and a family of morphisms $\alpha_i\colon D(i) \longrightarrow c$, for each object i of G, such that for each edge of the network, $e\colon i \longrightarrow i'$ it holds that $D(e); \alpha_{i'} = \alpha_i$.*

Definition 31 *A* colimiting cocone *(or colimit) $(c, \{\alpha_i\}_{i \in |G|})$ has the property that for any cocone $(d, \{\beta_i\}_{i\in|G|})$ there exists a unique morphism $\gamma\colon c \longrightarrow d$ such that $\alpha_i; \gamma = \beta_i$.*

By dropping the uniqueness condition and requiring only that a morphism γ should exist, a *weak* colimit is obtained.

When G is the category $\bullet \longleftarrow \bullet \longrightarrow \bullet$, G-colimits are called *pushouts.* When G is a discrete category (i.e. no arrows between objects other than identities), G-limits are called *coproducts.*

Definition 32 *A* cone *of a network $D : G \to C$ consists of an object c of C and a family of morphisms $\alpha_i\colon c \longrightarrow D(i)$, for each object i of G, such that for each edge of the network, $e\colon i \longrightarrow i'$ it holds that $\alpha_{i'} = \alpha_i; D(e)$.*

Definition 33 *A* limiting cone *(or limit) $(c, \{\alpha_i\}_{i\in|G|})$ has the property that for any cone $(d, \{\beta_i\}_{i\in|G|})$ there exists a unique morphism $\gamma\colon c \longrightarrow d$ such that $\gamma; \alpha_i = \beta_i$.*

When G is the category $\bullet \longrightarrow \bullet \longleftarrow \bullet$, G-limits are called *pullbacks.* When G is a discrete category, G-limits are called *products.*

P.2 Functors

Definition 34 *Let C and D be two categories. A* functor *$F : C \to D$ is a mapping that*

- *assigns to each object c of C an object $F(c)$ in D,*
- *assigns to each morphism $f \in C(c, d)$ a morphism $F(f) \in D(F(c), F(d))$ such that*
  - *$F(id_c) = id_{F(c)}$ for each $c \in |C|$,*
  - *$F(f; g) = F(f); F(g)$ for each $f \in C(a, b)$, $g \in C(b, c)$ and $a, b, c \in |C|$.*

Example For each category C, the identity functor $id_C : C \to C$ takes each object and each morphism to itself.

Example The forgetful functor F from the category of unsorted first-order ~~models~~ realizations of a signature F to *Set* takes each ~~model~~ realization M to the set M_U and each ~~model morphism~~ morphism of realizations $m : M \to N$ to its underlying function $m : M_U \to N_U$.

Example The covariant powerset functor $\mathcal{P} : \mathbb{S}et \to \mathbb{S}et$ maps each set A to the set of all subsets of A and each function $f : A \to B$ to the function that takes a subset X of A to the set $\{f(x) \mid x \in X\}$, which is a subset of B.

Example The covariant finite powerset functor $\mathcal{P}_{\mathit{fin}} : \mathbb{S}et \to \mathbb{S}et$ maps each set A to the set of all finite subsets of A and each function $f : A \to B$ to the function that takes a subset X of A to the set $\{f(x) \mid x \in X\}$, which is a subset of B.

[51] A network is called a diagram in category theory texts. This terminology is introduced to disambiguate OMS networks from UML diagrams.

[52] That is, it has a set of objects and sets of morphisms between them instead of classes.