PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 13-06

Institution-based Semantics for MOF and QVT-Relations

Daniel Calegari, Nora Szasz

2013

Institution- based semantics for MOF and QVT- relations
D. Calegari, N. Szasz
ISSN 0797-6410
Reporte Técnico RT 13-06
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay, 2013

Institution-based Semantics for MOF and QVT-Relations*

(Extended Version)

Daniel Calegari[1] and Nora Szasz[2]

[1] Facultad de Ingeniería, Universidad de la República, 11300 Montevideo, Uruguay
dcalegar@fing.edu.uy
[2] Facultad de Ingeniería, Universidad ORT Uruguay, 11100 Montevideo, Uruguay
szasz@ort.edu.uy

Abstract. In the Model-Driven Engineering (MDE) paradigm, software quality strongly depends on a (semi)automatic construction process driven by models and model transformations, which must be reliable and robust, since the tiniest error may grow and negatively impact in subsequent steps.

To cope with verification issues, a separation of duties between software developers is usually proposed. In general terms, MDE experts define models and transformations, while formal verification experts conduct the verification process. This view is often aided by (semi)automatic translations form the MDE elements to their formal representation in the semantic domain used by the verification experts. From a formal perspective, this requires semantic-preserving translations between the MDE elements and the chosen semantic domain.

The aim of this paper is to present formal semantics for the MetaObject Facility and the Query/View/Transformation Relations languages which are standard languages for defining metamodels and model transformation, respectively. The semantics are based on the Theory of Institutions and reflect the conformance relation between models and metamodels, and the satisfaction of transformation rules between pairs of models. The theory assists in the definition of semantic-preserving translations between institutions, specially between our institutions and other logics (first-order logic, rewriting logic, modal logic, etc.) which will be used for verification.

Keywords: MOF, QVT-Relations, formal semantics, Theory of Institutions, formal verification

* Technical Report / InCo-PEDECIBA / TR13-XX / ISSN 0797-6410 / Mayo 2013
http://www.fing.edu.uy/inco/pedeciba/bibpm/field.php/Main/ReportesT%e9cnicos

Semántica basada en Instituciones para MOF y QVT-Relations

(Versión Extendida)

Daniel Calegari[1] and Nora Szasz[2]

[1] Facultad de Ingeniería, Universidad de la República, 11300 Montevideo, Uruguay
dcalegar@fing.edu.uy
[2] Facultad de Ingeniería, Universidad ORT Uruguay, 11100 Montevideo, Uruguay
szasz@ort.edu.uy

Abstract. En el paradigma de Ingeniería Dirigida por Modelos (Model-Driven Engineering, MDE), la calidad del software depende fuertemente de un proceso de construcción (semi)automático guiado por modelos y transformaciones de modelos, que debe ser confiable y robusto, dado que el menor error puede crecer e impactar negativamente en pasos subsecuentes.

Para lidiar con aspectos de verificación, una separación de responsabilidades entre desarrolladores es usualmente propuesta. En términos generales, los expertos en MDE definen modelos y transformaciones, en tanto expertos en verificación formal conducen el proceso de verificación. Esta visión es generalmente asistida por traducciones (semi) automáticas de los elementos de MDE a la correspondiente representación formal en el dominio semántico utilizado por los expertos en verificación. Desde una perspectiva formal, esto requiere de de traducciones que preserven la semántica entre los elementos de MDE y el dominio semántico elegido.

El objetivo de este artículo es presentar una semántica formal para los lenguajes MetaObject Facility y Query/View/Transformation Relations que son lenguajes estándar para definir metamodelos y transformaciones de modelos, respectivamente. La semántica está basadas en la Teoría de Instituciones y refleja la relación de conformidad entre modelos y metamodelos, y la satisfacción de reglas de transformación entre pares de modelos. La teoría asiste en la definición de traducciones que preservan la semántica entre instituciones, especialmente entre nuestras instituciones y otras lógicas (lógica de primer orden, lógica de reescritura, lógica modal, etc.) que serán utilizadas para la verificación.

Palabras clave: MOF, QVT-Relations, semántica formal, Teoría de Instituciones, verificación formal

1 Introduction

The Model-Driven Engineering paradigm (MDE, [Ken02,Sch06]) envisions a software development life-cycle driven by models representing different views of the system to be constructed. Its feasibility is based on the existence of a (semi)automatic construction process driven by model transformations, starting from abstract models of the system and transforming them until an executable model is generated. The Object Management Group (OMG, [OMG]) has conducted a standardization process of languages for MDE. They defined the MetaObject Facility (MOF, [OMG03]) as the language for metamodeling as well as three transformation languages with different transformation approaches. In particular, the Query/View/Transformation Relations (QVT-Relations, [OMG09]) language follows a relational approach which consists on defining transformation rules as mathematical relations between source and target elements.

Since the quality of the whole development process strongly depends on the quality of the models and model transformations, verification is a must, and in some cases formal methods arise as a tool for strengthening verification results. To cope with this situation, a separation of duties between software developers is usually proposed. On the one side there are those experts in the MDE domain, and on the other, those in formal verification. This gives rise to different technological spaces [KBA02], i.e. working contexts with a set of associated concepts, body of knowledge, tools, required skills, and possibilities. In general terms, MDE experts define models and transformations, while formal verification experts conduct the verification process, often aided by some (semi)automatic generation process which translates the MDE elements to their formal representation in the semantic domain used for verification purposes.

We are exploring a comprehensive formal environment enabling this scheme. This environment requires semantic-preserving translations between the MDE elements and the chosen semantic domain. Moreover, different logics (e.g. modal logic, rewriting logic, predicate logic) can be used by verification experts. In this context, the biggest problem is perhaps the maintenance of multiple formal representations of the same MDE elements and the complexity of linking different semantic domains to perform a comprehensive verification using multiple semantic domains.

The aim of this paper is to present formal semantics for the MOF and the QVT-Relations languages in a flexible way to solve the problems described before. We base our proposal on the heterogeneous specification approach [CKTW08,Mos05], which consists in having different mathematical formalisms for expressing different parts of the overall problem and defining semantic-preserving mappings in order to allow "communication" between the formalisms. This approach uses as a basis the Theory of Institutions [GB84]. Using this theory we define institutions to represent the conformance relation between MOF models and metamodels and the satisfaction of QVT-Relations transformation rules between pairs of models. The theory also assists in the definition of semantic-preserving translations between our institutions and other logics (first-order logic, rewriting logic, modal logic, etc.) which will be used for verification.

Related Work There are many works defining the semantics of MOF and the conformance relation between models and metamodels, e.g. [BM09,CLST10,Fav09,SZ09]. These works usually define the semantics in terms of a shallow embedding of the language by providing a syntactic translation into another one (e.g. into first-order logic [SZ09] or rewriting logics [BM09]). We, on the contrary, do not want to depend on a general logic but to define a generic and minimal infrastructure allowing translations to other logics as needed. There are also some works with an algebraic/institutional approach. In [OW09] the authors propose an algebraic representation of metamodels based on many-sorted algebras, without an explicit representation of models within formulas. Moreover, in [BKMW08] the authors propose to define concrete institutions for any specific metamodel involved in a transformation. Unlike this work, we prefer to define a generic institution in order to specify transformations for any possible metamodel. Finally, in [CK08,JKMR13] the authors define institutions for simple and stereotyped UML Class Diagrams. We adapt those works for the purpose of defining the institution for the conformance relation.

With respect to QVT-Relations, there are also works (e.g. [ABK07,BHM09,LR11,SMR11]) defining the semantics of QVT-Relations in terms of a shallow embedding of the language by providing a syntactic translation into another one. As said before we do not follow this approach. In [BKMW08] transformations are represented as institution comorphisms, which is somehow restrictive since it assumes a semantic relation between metamodels. Finally, in [GdL12] the authors present a formal semantics for the QVT-Relations check-only scenario based on algebraic specification

and category theory. The definition of the institution is much more complex than ours and the work does not envision a scenario in which the elements of the transformation are translated to other logics for verification.

Organization The remainder of the paper is structured as follows. In Section 2 we introduce the elements involved in the MDE technical space which will be part of this work and we introduce a running example. Then, in Section 3 we summarize the general schema we follow for defining formal semantics based on the Theory of Institutions. In Section 4 we formally define an institution for MOF, and in Section 5 we define the institution for QVT-Relations. Then, in Section 6 we present an alternative definition of both institutions simplifying some of the definitions. Finally, in Section 7 we present some conclusions and guidelines for future work.

2 An Introduction to the MDE Technical Space

In MDE everything is a model, i.e. an abstraction of the system or its environment. Every model *conforms* to a metamodel, i.e. a model which introduces the syntax and semantics of certain kind of models. In the same way, a metamodel conforms to some metametamodel. A metametamodel is usually self-defined, which means that it can be specified by means of its own semantics. Model transformations (or just transformations from now on) can also be considered as models conforming to a transformation metamodel. A transformation basically takes as input a model conforming to certain metamodel and produces as output another model conforming to another metamodel (possibly the same). This very simple transformation schema is summarized in Figure 1, and can be extended to consider bidirectional transformations or to take more than one source model as input and/or produce multiple target models as output, among other extensions, as studied in [CH06].

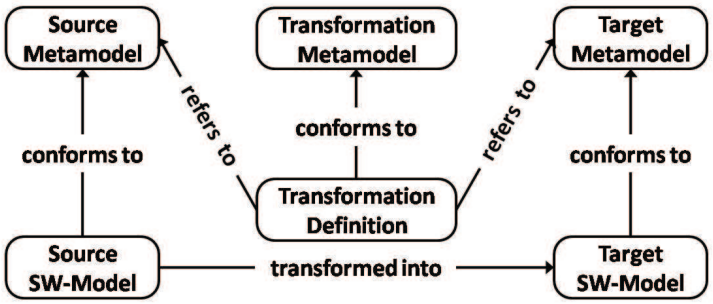


Fig. 1: An overview of a model transformation

The MetaObject Facility (MOF, [OMG03]) is a standard language for metamodeling, which has a close relation with UML Class Diagrams [OMG05]. In few words, a metamodel defines classes which can belong to a hierarchical structure and some of them must be defined as abstract (there are no instances of them). Any class has properties which can be attributes (named elements with an associated type which can be a primitive type or another class) and associations (relations between classes in which each class plays a role within the relation). Every property has a multiplicity constraining the number of elements that can be related through the property.

Besides a metamodel usually defines a modeling language which has a concrete syntax, it is possible to represent a model using the same languages as for metamodels. Moreover, for representing models (which are a kind of "instance" of a metamodel) and instances of models, there is the graphical representation provided by UML Object Diagrams [OMG05]. In some cases, there are conditions (called invariants) that cannot be captured by the structural rules of these languages, in which case modeling languages are supplemented with another logical language, e.g. the Object Constraint Language (OCL, [OMG10]).

Let us consider the following example which is a simplified version of the well-known Class to Relational transformation [OMG09]. The metamodel on the left side of Figure 2 defines UML class diagrams, where classifiers (classes and primitive types as string, boolean, integer, etc.) are contained in packages. Classes can contain one or more attributes and may be declared as persistent, whilst attributes have a type that is a primitive type. On the other side, relational models conform to the metamodel on the right side of Figure 2. Every schema contains a number of tables and each table has a number of columns. Each column has a name and a kind, which can be the primary keys of the corresponding table.

With respect to model transformations, the OMG defines three languages with different transformation approaches. In particular, the Query/View/Transformation Relations (QVT-Relations, [OMG09]) follows a relational approach which consists on defining transformation rules as mathematical relations between source and target elements. A transformation can be viewed as a set of interconnected relations which are of two kinds: top-level and non-top-level. Top-level relations must hold in any transformation execution whilst non-top-level ones are required to hold only when

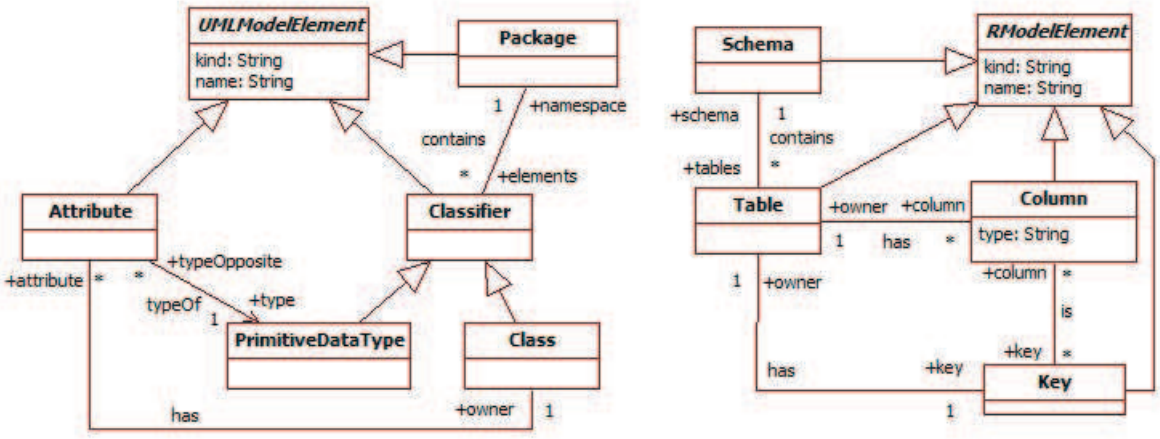


Fig. 2: Source and target metamodels of the example

they are referred from another relation. For the purpose of this work we can view a relation as having the following abstract structure [OMG09]:

```
[top] relation R
{
   <R_var_set> <R_par_set>

    Domain
    {
        <domain_k_var_set>
        <domain_k_pat>
    } //k = 1,2

    [when <when_var_set> <when_cond>]
    [where <where_cond>]
}
```

where:

- `<R_var_set>` is the set of variables occurring in the relation.
- `<domain_k_var_set>` $\subseteq$ `<R_var_set>` is the set of variables occurring in domain k (k = 1,2).
- `<when_var_set>` $\subseteq$ `<R_var_set>` is the set of variables occurring in the `when` clause.
- `<R_par_set>` $\subseteq$ `<R_var_set>` is the set of variables taken as parameters
- `<domain_k_pat>` is the pattern to be checked in domain the k (k = 1,2)
- `<when_cond>` and `<where_cond>` represent the `when` and `where` clauses

Although transformations can be defined between multiple metamodels at the same time, in this work we will only consider a source and a target metamodel. We neither considered:

- black-box operations (there is an advanced feature not commonly used in practice)
- rule and transformation overriding (there are advanced features not commonly used in practice)
- auxiliary functions and queries (there is syntactic sugar)
- keys definition (there are used for object creation not within the checking semantics)

A pattern is used to find matching sub-graphs in a model and can be viewed as a graph, where typed pattern elements are the nodes of the graph and pattern links are the edges, together with a `predicate` which is a boolean expression. The predicate may refer to variables other than the pattern elements (those in `<domain_k_var_set>`); these are the free variables of a pattern. A pattern can be represented as follows.

```
Pattern =
{
    e1: <classname1>, e2: <classname2> ... en:<classnameN>
    l1 : <assoc1> (ei, ej) ... lm:<assocM>(eu, ew)
    where <predicate>
}
```

We simplfy the pattern structure by not considering:

- opposite roles in object templates (they can be expressed as conditions within a template)
- collection templates (they are advanced features not commonly used in practice)

A `when` clause specifies the conditions under which the relationship needs to hold, whilst the `where` clause specifies the condition that must be satisfied by all model elements participating in the relation, and it may constrain any of the variables in the relation and its domains. The `when` and `where` clauses, as well as the `<predicate>` of a pattern, may contain arbitrary boolean OCL expressions in addition to the relation invocation expressions.

Finally, any relation can define a set of primitive domains which are data types used to parameterize the relation. In this sense, top-level relations can be parametric when called from a `when` clause, whereas non-top-level relations are always parametric since they are called for given source and target domains elements.

We customize the standard checking semantics and take the first and second patterns as the source and target patterns, respectively. Thus, a rule holds if for each valid binding of variables of the `when` clause and variables of domains other than the target domain, that satisfy the `when` condition and source domain patterns and conditions, there must exist a valid binding of the remaining unbound variables of the target domain that satisfies the target domain pattern and `where` condition. More formally, using `|<var_set>|` as a binding of variables of the set `<var_set>`, and `<exc_domain_k_var_set>` as the variables of domain k that do neither occur in the other domain nor the `when` clause, the rule holds if:

$$
\begin{array}{l}
\forall\, |< \text{when\_var\_set} >|, \\
\quad (\, < \text{when\_cond} > \rightarrow \\
\qquad \forall\, |< \text{R\_var\_set} > \backslash (< \text{when\_var\_set} > \cup < \text{exc\_domain\_2\_var\_set} >)|, \\
\qquad\quad (\, < \text{domain\_1\_pat} > \rightarrow \\
\qquad\qquad \exists\, |< \text{exc\_domain\_2\_var\_set} >|, \\
\qquad\qquad\quad (\, < \text{domain\_2\_pat} > \wedge < \text{where\_cond} >)))
\end{array}
$$

The example transformation basically describes how persistent classes within a package are transformed into tables within a schema. Attributes of a class are transformed into columns of the corresponding table, and the primary key is defined by default. Below we show the transformation specification defined in QVT-Relations.

```
transformation umlToRdbms(uml:SimpleUML, rdbms:SimpleRDBMS)
{
  top relation PackageToSchema {
    pn: String;
    domain uml p:Package {name=pn};
    domain rdbms s:Schema {name=pn};
  }

  top relation ClassToTable {
    cn, prefix: String;
    domain uml c:Class {namespace=p:Package {},
       kind='Persistent', name=cn};
    domain rdbms t:Table {schema=s:Schema {}, name=cn,
       column=cl:Column {name=cn+'_tid', type='NUMBER'},
       key=k:Key {name=cn+'_pk', column=cl}};
    when {
      PackageToSchema(p, s);
    }
    where {
      prefix = '';
      AttributeToColumn(c, t, prefix);
    }
  }

  relation AttributeToColumn {
    an, pn, sqltype: String;
    domain uml c:Class {attribute=a:Attribute {name=an,
       type=p:PrimitiveDataType {name=pn}}};
    domain rdbms t:Table {column=cl:Column {name=cn,
       type=sqltype}};
    primitive domain prefix:String;
    where {
       cn = if (prefix = '') then an else prefix+'_'+an endif;
       sqltype = if (pn='INTEGER')
                 then 'NUMBER'
                 else if (pn='BOOLEAN')
                      then 'BOOLEAN' else 'VARCHAR'
                      endif
                 endif;
     }
  }
}
```

In Figure 3 there is an example of a source model and its corresponding target model. The source model is composed by a persistent class of name `ID` within a package of name `Package`. The class has an attribute of name `value` and type `String` which is a primitive type. The forward execution of the transformation gives the target model which contains a schema of name `Package` with a table of name `ID`, which corresponds to the package and class in the source model. The table has two columns, one of name `value` and type `VARCHAR` corresponding to the string attribute in the source class, and another which is a default primary key without any correspondence in the source model.

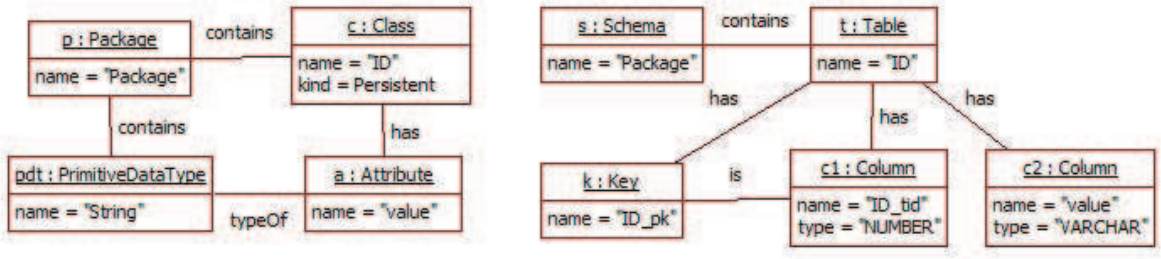


Fig. 3: Source and target models for the example

3 An Environment for Verification

We are exploring a comprehensive formal environment for the formal verification of different aspects of a model transformation using heterogeneous verification approaches [CS13]. The environment is based on representing models (from now on SW-models), metamodels, the conformance relation, transformations and verification properties in some consistent and interdependent way following the heterogeneous specification approach [CKTW08,Mos05].

This approach is based on providing *Institutions* for the languages which are part of the environment. The concept of Institution [GB84] was originally introduced to formalize the notion of logical system. In fact, many different logics as first-order, higher-order, modal, rewriting, among others have been shown to be institutions. Informally, an institution consists of a collection of signatures (vocabularies for constructing sentences in a logical system), signature morphisms (allowing many different vocabularies at once), a collection of sentences and models (providing semantics) for a given signature, and a satisfaction relation of sentences by models, such that when signatures are changed (by a signature morphism), satisfaction of sentences by models changes consistently.

The formal definition of an institution relies on Category Theory [Lan98]. As defined in [ST12], an institution consists of:

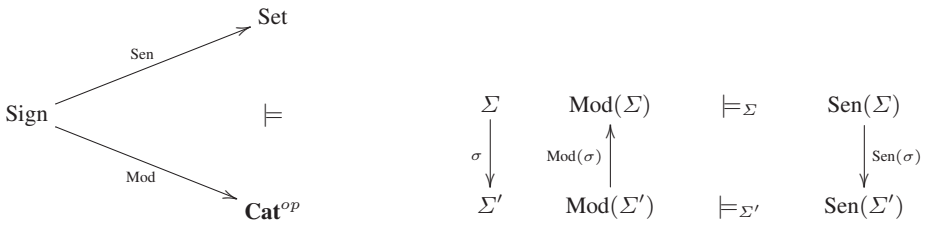
1. a category Sign of *signatures*;
2. a functor Sen : Sign $\rightarrow$ Set, giving a set Sen(Σ) of Σ-*formulas* for each signature $\Sigma \in$ |Sign| [3] and a function Sen(σ):Sen(Σ_1)$\rightarrow$ Sen(Σ_2) translating Σ_1-formulas to Σ_2-formulas for each signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$;
3. a functor Mod : Sign $\rightarrow$ $\mathbf{Cat}^{op}$, giving a category Mod(Σ) of Σ-*models* for each signature $\Sigma \in$ |Sign| and a functor Mod(σ):Mod(Σ_2)$\rightarrow$ Mod(Σ_1) translating Σ_2-models to Σ_1-models (and Σ_2-morphisms to Σ_1-morphisms) for each signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$;
4. for each signature $\Sigma \in$ |Sign|, a *satisfaction relation* $\models_\Sigma \subseteq$ |Mod(Σ)| $\times$ Sen(Σ);

such that for any signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$ the translation Mod(σ) of models and Sen(σ) of formulas preserve the satisfaction relation, that is, for any $\varphi \in$ Sen(Σ) and $M_2 \in$ |Mod(Σ_2)|:

$$M_2 \models_{\Sigma_2} \text{Sen}(\sigma)(\varphi) \text{ iff } \text{Mod}(\sigma)(M_2) \models_{\Sigma_1} \varphi$$

In Figure 4 there is a graphical representation of these elements and their relations. In the left side there is a representation of the categories defined for signatures (Sign), formulas (Set) and models ($\mathbf{Cat}^{op}$) and the functors relating them, as well as the satisfaction relation which relates formulas and models. In the right side there is a signature morphism σ allowing a change of notation between signatures Σ and Σ'. Sentences translate in the same direction as the change of notation, whereas models translate in the opposite direction. Because reversing the direction of morphisms gives a contravariant functor, the definition below uses $\mathbf{Cat}^{op}$, the opposite of the category of categories. The satisfaction condition states that *truth is invariant under change of notation* [GB92].

[3] |C| is the collection of objects of a category C



(a) Categories and functors

(b) The satisfaction condition

Fig. 4: Graphical view of an institution

The notion of an institution can be used to represent any specification language since it provides ways of representing the syntax and semantics of the language, as well as the relation between them by means of a satisfaction relation between them, as in [CKTW08]. In this work we provide an institution for QVT-Relations check-only unidirectional transformations. This kind of transformations only checks if a target model is the result of transforming the source SW-model according to the transformation rules. This institution needs a representation of SW-models and metamodels, therefore we first define an institution for MOF for expressing the conformance relation between them.

In order to use our institutions for verification purposes, there are two alternatives. The first one is to extend the institutions from a proof-theoretic point of view by defining a *logic*. A logic $\mathcal{LOG} = (\text{Sign}, \text{Sen}, \text{Mod}, \models, \vdash)$ is an institution $(\text{Sign}, \text{Sen}, \text{Mod}, \models)$ equipped with an *entailment system* $\vdash$ that is, a relation $\vdash_\Sigma \subseteq P(\text{Sen}(\Sigma)) \times \text{Sen}(\Sigma)$ for each $\Sigma \in \text{Sign}$, such that some properties are satisfied. Particularily, the entailment system must be sound, i.e. $\Psi \vdash_\Sigma \varphi$ implies $\Psi \models_\Sigma \varphi$. In some cases the entailment system can also be complete, i.e. $\Psi \models_\Sigma \varphi$ implies $\Psi \vdash_\Sigma \varphi$.

The second alternative is to formally translate our institutions into another logic. This can be done through *institution comorphisms* [GR02], which capture how a *weaker* and *poorer* institution can be represented in a *stronger* and *richer* one. The importance of comorphisms is such that it is possible (in some cases) to re-use (*borrow*) the entailment systems of an institution in order to prove properties. As pointed out in [Mos05], "if we have a sound proof calculus for entailment in $\mathcal{J}$, and if we have an institution comorphism $\rho : \mathcal{I} \to \mathcal{J}$ admitting borrowing of entailment for $\mathcal{I}$, we can use the proof calculus also for proving entailment concerning $\mathcal{I}$-specifications in $\mathcal{J}$: we just have to translate our proof goals."

We will take the second alternative by defining comorphisms from our institutions to a host logic and supplement this information with properties specified in the host logic. In particular, we are in the process of defining a comorphism to the Common Algebraic Specification Language (CASL, [MHST03]), a general-purpose specification language. The institution underlying CASL is the sub-sorted partial first-order logic with equality and constraints on sets $SubPCFOL^=$, a combination of first-order logic and induction with subsorts and partial functions. The importance of CASL is that it is the main language within the Heterogenous Tool Set (Hets, [Mos05]), which is a tool meant to support heterogeneous multi-logic specifications. Hets allows defining the institutions and comorphisms, and also provides proof management capabilities for monitoring the overall correctness of a heterogeneous specification whereas different parts of it are verified using (possibly different) proof systems. Hets already supports several interconnected logics which is shown in Figure 5.

To the best of our knowledge, Hets does not support the MDE paradigm, i.e. it does not have specific languages for the specification of MDE elements. We plan to include our institutions as logics in Hets, in such a way that a developer can import a transformation, use Hets to specify additional verification properties which must be addressed, and perform the verification assisted by the tool. We can use CASL together with some of the other logics in the graph of logics currently supported by Hets.

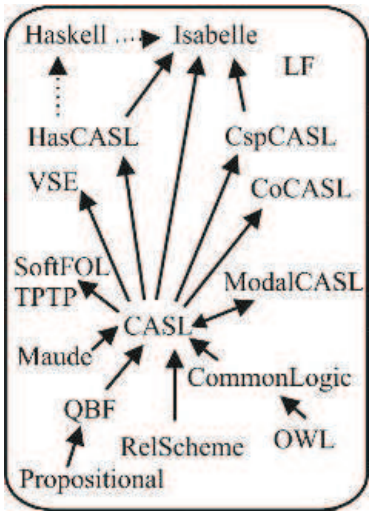


Fig. 5: Basic graph of logics and logic translations within Hets

3.1 Defining the Institutions

In Section 4 we define the institution for the conformance relation for a generic metamodeling language equivalent to MOF, basing our proposal on the institution defined for UML class diagrams in [CK08,JKMR13]. We adapt the definitions with the purpose of representing metamodels. Unlike [CK08], in our definition:

- there are no derived relations (not used in transformations),
- the signature has an explicit representation of abstract classes,
- the signature has an explicit representation of datatypes,
- there are only 2-ary properties (associations and attributes),

Moreover, unlike MOF, we do not consider aggregation, uniqueness and ordering properties within a property end, operations on classes, or packages. Properties and operations are not commonly used within transformations, whereas packages are just used for organizing metamodel elements. We also use an explicit syntactic representation of SW-models following the schema introduced in [SZ09], which is shown in Figure 6. This allows using concrete SW-models information for verification purposes, as we will see later.

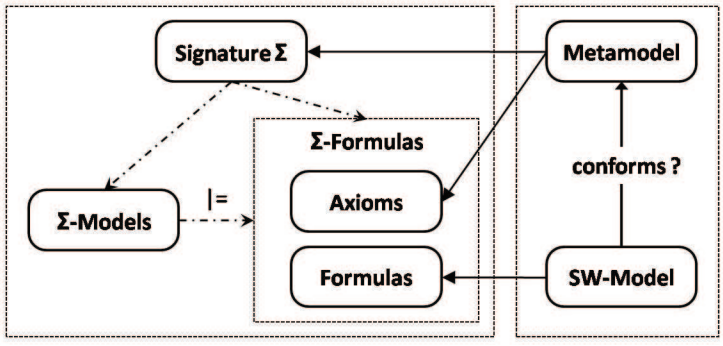


Fig. 6: The conformance relation as an institution

From any metamodel we can derive a signature with a representation of types and properties (attributes and associations), and a set of axioms stating invariants which must hold on every conforming SW-model. Up to now we have considered multiplicity constraints. However, it will be possible to add other kind of constraints through comorphisms as will be explained later on. Formulas are composed by these axioms and a representation of a potentially conforming SW-model. In few words, any formula represents a SW-model structurally conforming to the metamodel and the set of constraints that must apply in order to achieve full conformance.

Any institution model (from now on just model) is a semantic representation of a potentially conforming SW-model. The model is composed by objects and relations between them. However, we want to prove conformance with respect to the formula representing the SW-model. In this sense, we can reduce the model with respect to the formula such that elements and properties in such model are those with a corresponding element in the formula.

This allows us to define the satisfaction relation answering the question: does the SW-model conform to the metamodel?. The model satisfies the corresponding formula if: (a) the SW-model has a correspondence with the model (isomorphic with respect to the reduced model), and (b) the reduced model satisfies the multiplicity constraints.

In Section 5 we also define an institution for QVT-Relations check-only unidirectional transformations. Any transformation can be viewed as a set of interconnected relations which must hold. For the definition of this institution we follow the schema shown in Figure 7.

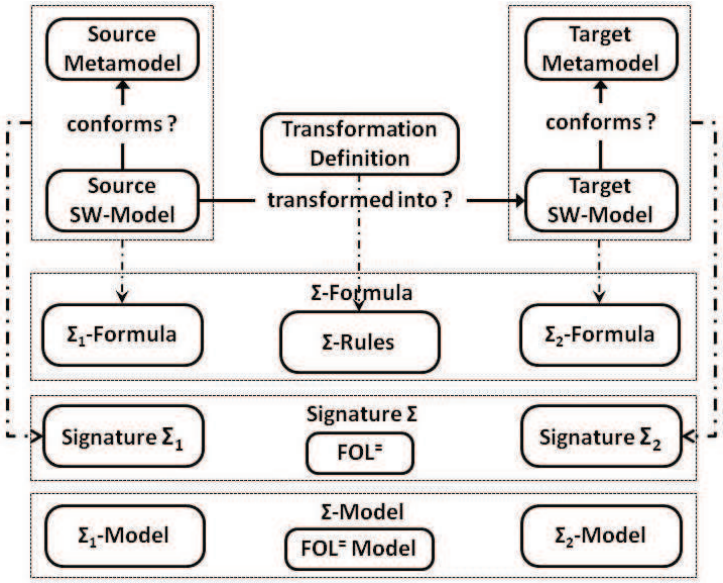


Fig. 7: A model transfromation as an institution

The institution takes the institutional representation of the source and target elements and supplements the formulas with a representation of the transformation rules. In this case, the satisfaction relation also answers the question: is the target SW-model the result of transforming the source SW-model according to the transformation rules? For defining the satisfaction relation we use the standard checking semantics defined in [OMG09].

Finally, in Section 6 we present an alternative approach to the definition of both institutions. We basically change the definition of signatures in the MOF institution in close relation with the ideas presented in [JKMR13], in which instances (class objects and type values) are represented within the signature instead of within the formulas. This change simplifies several definition.

The Object Constraint Language As mentioned before, the `when` and `where` clauses, as well as the `<predicate>` of a pattern, may contain arbitrary boolean OCL expressions. From a formal perspective we would rather have an institution for OCL which would allow us to use the language not only for constraining the transformation rules, but also for expressing general constraints on metamodels and transformations. Unfortunately there is no institution for OCL, and defining one is not an easy task, so is left for future work.

In our examples we consider an institution for first-order logic with equality ($FOL^{=}$) as defined in [ST12]. With this decision we are not losing expressive power (there are works [BKS02] with the aim of expressing OCL into first-order logic). In $FOL^{=}$, signatures are many-sorted algebraic signatures enriched with predicate symbols of the form (S, Ω, Π) where S is a set (of sort names), $\Omega = (\Omega_{w,s})_{w \in S^*, s \in S}$ is a family of sets (of operation names with their arities and result sorts indicated just as in algebraic signatures) and $\Pi = (\Pi_w)_{w \in S^*}$ is a family of sets (of predicate or relation names with their arities indicated). Signature morphisms are as usual between elements in the signatures. Moreover, sentences are first-order formulae built out from atomic formulas using the standard propositional connectives ($\wedge$,$\vee$,$\Rightarrow$,$\Leftrightarrow$,$\neg$) and quantifiers ($\forall$, $\exists$). The atomic formulae are equalities of the form $t = t'$, where t and t' are (S,Ω)-terms (possibly with variables) of the same sort, atomic predicate formulae of the form $p(t_1, ..., t_n)$, where $p \in \Pi_{s_1 ... s_n}$ and $t_1, ..., t_n$ are terms (possibly with variables) of sorts $s_1, ..., s_n$, respectively, and the logical constants true and false. Models are many-sorted first-order structures, i.e. consisting of a carrier set $|D|_s$ for each sort name $s \in S$, a function f_D for each operation name $f \in \Omega$, and a relation p_D for each $p \in \Pi$. Finally, the satisfaction relation is the usual satisfaction of a first-order sentence in a first-order structure. The formulas can also include variables $X^s = (X^s)_{s \in S}$, so for the satisfaction relation we consider variable assignments $\mu_s : X^s \to |D|_s$ for each $s \in S$.

Recursive Model Transformations We need to forbid cycles of rule invocations to avoid infinite recursion. Notice that `when` and `where` clauses conform a potentially cyclic graph of dependencies between transformation rules. Cycles are however not problematic unless the satisfaction of a rule involving a set of SW-model elements depends recursively on its own satisfaction. In this case we have infinite recursion which cannot be handled by our proposal. We thus assume that recursion is well-founded, i.e. no rule will be called twice in the same chain of dependencies for the same set of elements. This constraint ensures well-foundness since we always have a finite set of element in any SW-model. Another alternative evaluated in [GdL12] is to forbid cycles of dependencies to avoid infinite recursion. However, this alternative is too restrictive in practice.

4 An Institution for MOF

In this section we present the formal definition of an institution $\mathcal{I}^{\mathrm{C}}$ for the conformance relation for a generic metamodeling language equivalent to MOF. As said before, this definition is based on the institution for UML class diagrams defined in [CK08], but adapted for representing metamodels. Along the definition we will illustrate the concepts introduced with the example presented in Section 2.

4.1 Signatures and Formulas

A class hierarchy is a partial order $\boldsymbol{C} = (C, \leq_C)$ where C is a set of class names, and $\leq_C \subseteq C \times C$ is the subclass (inheritance) relation. By $\boldsymbol{T}(\boldsymbol{C})$ we denote the *type extension* of $\boldsymbol{C}$ by primitive types and type constructors. $\boldsymbol{T}(\boldsymbol{C})$ is likewise a class hierarchy $(T(C), \leq_{T(C)})$ with $C \subseteq T(C)$ and $\leq_C \subseteq \leq_{T(C)}$. As in [JKMR13], in order to provide generic access to primitive types, like Boolean, and String, we treat these as built-ins with a standard meaning (they must defined within $T(C)$). All other classes are assumed to be inhabited, i.e., to contain at least one object. However, unlike [JKMR13] in which is assumed the existence of an object $null$, we impose that if $c \in C|_{abstract}$ then there exists another $c' \in T(C)$ in the hierarchy such that $c' \leq_{T(C)} \ldots \leq_{T(C)} c$ and $c' \notin C|_{abstract}$ having at least one object.

As we mentioned before, from a metamodel we can derive a signature $\Sigma = (\boldsymbol{T}, \boldsymbol{P})$ declaring:

- a type extension of a finite class hierarchy $\boldsymbol{T} = (T(C), \leq_{T(C)}, C|_{abstract})$ extended with a subset $C|_{abstract} \subseteq C$ denoting abstract classes, and
- a properties declaration (attributes and associations) $\boldsymbol{P} = (R, P)$ where R is a finite set of role names and P is a finite set $(p_w)_{w \in (R \times T(C)) \times (R \times T(C))}$ of property names indexed over pairs of a role name and a class (or type) name, such that for any class or type name $c \in C$, the role names of the properties in which any $c' \leq_{T(C)} c$ is involved are all different[4]. If $p_w \in P$ with $w = ((r_1, c_1)(r_2, c_2))$, we write $p(r_1 : c_1, r_2 : c_2) \in P$.

Example

From the class metamodel in Figure 2 we can derive a signature $(\boldsymbol{T}, \boldsymbol{P})$ such that $\boldsymbol{T}$ is a type extension based on a class hierarchy trivially derived from the classes and hierarchical relations within the metamodel, and extended with the built-in type `String`.

$$T(C) = \{\text{UMLModelElement}, \text{Package}, ..., \text{String}\}$$
$$\leq_{T(C)} = \{\text{Package} \leq_{T(C)} \text{UMLModelElement}, ...\}$$
$$C|_{abstract} = \{\text{UMLModelElement}\}$$

$$\boldsymbol{P} = (R, P) \text{ where}$$

$$R = \{\text{namespace}, \text{elements}, \text{type}, \text{typeOpposite}, ...\}$$

$$P = \{\text{contains}(\text{namespace} : \text{Package}, \text{elements} : \text{Classifier}),$$
$$\text{name}(\text{UMLModelElement} : \text{UMLModelElement}, \text{name} : \text{String}),$$
$$\text{typeOf}(\text{typeOpposite} : \text{Attribute}, \text{type} : \text{PrimitiveDataType}), ...\}$$

[4] Formally, if $p(r_1 : c_1, r_2 : c_2)$ and $q(s_1 : d_1, s_2 : d_2)$ are properties in P and $c_k = d_l \in T(C)$, then $r_i \neq s_j$ for any $i \neq k$ and for any $j \neq l$ $(1 \leq i \leq 2, 1 \leq j \leq 2)$

From a metamodel, it is also possible to derive a set of formulas constraining the set of SW-models conforming to it. Unlike [CK08], the set of formulas is complemented with the representation of a SW-model. In few words, any formula represents a SW-model structurally conforming to the metamodel and the set of constraints that must apply in order to achieve full conformance. Formally, given a signature as defined before, and variables $(X^c)_{c \in T(C)}$ verifying $X^{c_1} \subseteq X^{c_2}$, if $c_1 \leq_{T(C)} c_2$ (i.e. in accordance with the inheritance relation), the set Φ of Σ-formulas is defined by

$$
\begin{aligned}
\Phi &::= (Obj^*, Rel^*, \phi^*) \\
Obj &::= z^e \\
Rel &::= \texttt{rel}(p_w, x^c, y^d) \\
\phi &::= \#\Pi = n \mid n \leq \#\Pi \mid \#\Pi \leq n \\
\Pi &::= R \bullet P
\end{aligned}
$$

where $c, d, e \in T(C)$, $x^c, y^d \in Obj^*$, $x^c \in X^c$, $y^d \in X^d$, $z^e \in X^e$, $p_w \in P$, $w = (r_1, c)(r_2, d)$, $n \in \mathbb{N}$. We use $\bullet$ as the select/partition operator in Π representing the selection of the elements in the opposite side of role R in property P.

A SW-model is represented by a set Obj^* of variables indexed by class and type names which are a syntactic representation of object and values, together with a set of relations connecting these elements. A relation of the form $\texttt{rel}(p_w, x^c, y^d)$ is well-typed and states that the elements x^c and y^d are related through the property p_w. Up to now we have only considered multiplicity constraints. However, it is possible to add other kind of constraints through comorphisms, as explained before.

Example

The formula $\varphi = (Obj, Rel, \phi)$ corresponding to the axioms extracted from the metamodel plus the SW-model in Figure 3 is defined by

- Obj = $\{\mathrm{p}, \mathrm{c}, \mathrm{a}, \mathrm{pdt}, \mathrm{Package}, \mathrm{Persistent}, \mathrm{String}, \mathrm{ID}, \mathrm{value}, \mathrm{NULL}\}$ with each of these elements indexed by its corresponding class and type names
- Rel = $\{\mathrm{rel}(\mathrm{contains}, \mathrm{p}, \mathrm{c}), \mathrm{rel}(\mathrm{contains}, \mathrm{p}, \mathrm{pdt}), \mathrm{rel}(\mathrm{has}, \mathrm{c}, \mathrm{a}), \mathrm{rel}(\mathrm{type}, \mathrm{a}, \mathrm{pdt}), \mathrm{rel}(\mathrm{name}, \mathrm{p}, \mathrm{Package}), \mathrm{rel}(\mathrm{kind}, \mathrm{c}, \mathrm{Persistent}), \mathrm{rel}(\mathrm{name}, \mathrm{pdt}, \mathrm{String}), ...\}$
- ϕ = $\{\#(\mathrm{UMLModelElement} \bullet \mathrm{name}) = 1, \#(\mathrm{UMLModelElement} \bullet \mathrm{kind}) = 1, \#(\mathrm{elements} \bullet \mathrm{contains}) = 1, \#(\mathrm{attribute} \bullet \mathrm{has}) = 1, ...\}$.

Let $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i)$ $(i = 1, 2)$ with $\boldsymbol{T}_i$ = $(T(C_i), \leq_{T(C_i)}, C_i\,|_{abstract})$ and $\boldsymbol{P}_i$ = (R_i, P_i). A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ is a triple of maps $\langle \sigma_T, \sigma_R, \sigma_P \rangle$ between class names, role names, and property names, such that the following conditions hold:

- $a \in C_1$ implies $\sigma_T(a) \in C_2$,
- $a \in T(C_1) \backslash C_1$ implies $\sigma_T(a) \in T(C_2) \backslash C_2$,
- $a, b \in T(C_1)$ with $a \leq_{T(C_1)} b$ implies $\sigma_T(a) \leq_{T(C_2)} \sigma_T(b)$,
- $a \in C_1|_{abstract}$ implies $\sigma_T(a) \in C_2|_{abstract}$,
- $p_w \in P_1$ implies $\sigma_P(p)_{\sigma(w)} \in P_2$, where σ is the canonical extension of σ_T and σ_R to words in $(R \times T(C))(R \times T(C))$.

Given a set of variables $X_2 = (X_2^{c_2})_{c_2 \in T(C_2)}$, we define a set $X_2\,|_\sigma = X_1 = (X_1^{c_1})_{c_1 \in T(C_1)}$ by $X_1^{c_1} = X_2^{\sigma(c_1)}$. Signature morphisms extend to formulas over Σ_1 and $X_2|_\sigma$ as follows. Given a Σ_1-formula $\varphi = (Obj, Rel, \Phi)$, $\sigma(\varphi)$ is the canonical application of the signature morphism to every element in the sets Obj, Rel and Φ such that:

- $\sigma(x^c) = x^{\sigma(c)}$
- $\sigma(\texttt{rel}(p_w, x^c, y^d)) = \texttt{rel}(\sigma_P(p)_{\sigma(w)}, x^{\sigma(c)}, y^{\sigma(d)})$
- $\sigma(r \bullet p) = \sigma_R(r) \bullet \sigma_P(p)$

As stated in [CK08], it is easy to show that the composition of signature morphisms is a signature morphism, that composition is associative, and identities are signature morphisms. Thus, signatures and signature morphisms define a category. Moreover, there is a functor Sen giving a set of formulas for each signature and a function translating sentences for each signature morphism. Proofs of these results can be found in Appendix A.

4.2 Models

Given a class hierarchy $\boldsymbol{C} = (C, \leq_C)$, a $\boldsymbol{C}$-object domain $\boldsymbol{O}$ is a family $(O_c)_{c \in C}$ of sets of object identifiers verifying $O_{c_1} \subseteq O_{c_2}$ if $c_1 \leq_C c_2$. Given moreover a type extension $\boldsymbol{T}$, the *value extension* of a $\boldsymbol{C}$-object domain $\boldsymbol{O} = (O_c)_{c \in C}$ by primitive values and value constructions, which is denoted by $\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O})$, is a $\boldsymbol{T}(\boldsymbol{C})$-object domain $(V_c)_{c \in T(C)}$ such that $V_c = O_c$ for all $c \in C$. We consider disjoint sets of objects within the same hierarchical level, in particular, if $c_1 \leq_C c$ and $c_2 \leq_C c$, then $O_{c_1} \cap O_{c_2} = \emptyset$.

From now on, let us consider a fixed signature $\Sigma = (\boldsymbol{T}, \boldsymbol{P})$ with $\boldsymbol{T} = (T(C), \leq_{T(C)}, C|_{abstract})$ and $\boldsymbol{P} = (R, P)$.

A Σ-interpretation $\mathcal{I}$ consists of a pair $(\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}), \boldsymbol{A})$ where

- $\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}) = (V_c)_{c \in T(C)}$ is a $\boldsymbol{T}(\boldsymbol{C})$-object domain such that $c_2 \in C|_{abstract}$ implies $O_{c_2} = \bigcup_{c_1 \leq_C c_2} O_{c_1}$, and
- $\boldsymbol{A}$ contains a relation $p^{\mathcal{I}} \subseteq V_{c_1} \times V_{c_2}$ for each relation name $p(r_1 : c_1, r_2 : c_2) \in P$ with $c_1, c_2 \in T(C)$

Example

An interpretation $\mathcal{I}$ can be defined as follows:

- A $\boldsymbol{T}(\boldsymbol{C})$-object domain consisting of

$$V_{\text{Class}} = \{c1, c2\}$$
$$V_{\text{PrimitiveDataType}} = \{pdt1, pdt2\}$$
$$V_{\text{Classifier}} = V_{\text{Class}} \cup V_{\text{PrimitiveDataType}}$$
$$V_{\text{Package}} = \{p1\}$$
$$V_{\text{Attribute}} = \{a1, a2, a3\}$$
$$V_{\text{UMLModelElement}} = V_{\text{Classifier}} \cup V_{\text{Package}} \cup V_{\text{Attribute}}$$
$$V_{\text{String}} = \{Pac, Str, Per, nul, ID, val\}$$

- A set $\boldsymbol{A}$ consisting of relations:

$$\text{contains}^{\mathcal{I}} = \{(p1, c1), (p1, c2)(p1, pdt1), (p1, pdt2)\}$$
$$\text{name}^{\mathcal{I}} = \{(p1, Pac), (c1, ID), (c2, nul), (a1, val), ...$$
$$\text{kind}^{\mathcal{I}} = \{(p1, nul), (c1, Per), (c2, Per), ...$$
$$\text{type}^{\mathcal{I}} = \{(a1, pdt1), (a2, c2), (a3, pdt1)\}$$
$$...$$

Given variables $X = (X^c)_{c \in T(C)}$, a *valuation* $\beta = (\beta^c)_{c \in T(C)}$ assigns values to variables, i.e., $\beta^c : X^c \to V^c$ for every $c \in T(C)$ such that:

- for every $x^c \in X^c$ with $c \in C$, $\beta^c(x) \in O^c$
- for every $x^c \in X^c$ with $c \in T(C) \backslash C$, $\beta^c(x) \in V^c \backslash O^c$
- for every $x^c, y^c \in X^c$, $\beta^c(x) \neq \beta^c(y)$

We can extend the valuation for sets, i.e. given a set $S \subseteq \bigcup_{c \in T(C)} X^c$, $\beta(S) = \{\beta^c(x) \mid x \in S\}$.

Example

We can define a valuation β^c such that:

$$\beta^{\text{Package}}(\text{p}) = p1$$
$$\beta^{\text{Class}}(\text{c}) = c$$
$$\beta^{\text{Attribute}}(\text{a}) = a$$
$$\beta^{\text{PrimitiveDataType}}(\text{pdt}) = pdt$$
$$\beta^{\text{String}}(\text{Package}) = Pac$$
$$\beta^{\text{String}}(\text{ID}) = ID$$
$$\beta^{\text{String}}(\text{Persistent}) = Per$$
$$\beta^{\text{String}}(\text{String}) = Str$$
$$\beta^{\text{String}}(\text{value}) = val$$
$$\beta^{\text{String}}(\text{NULL}) = nul$$

Given a Σ-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$, the interpretation evaluates relations as follows: if $p(r_1 : c_1, r_2 : c_2)$ then $(r_i \bullet p)^{\mathcal{I}} = \{\{t \in p^{\mathcal{I}} \mid \pi_i(t) = o\} \mid o \in V_{c_i}\}$ $(i = 1, 2)$. The evaluation $(r_i \bullet p)^{\mathcal{I}}$ gives a set of sets of pairs of semantic elements connected through property p, grouped by the semantic elements having role r_i. Note that this set can be empty if the element with role r_i is not connected with any one.

Example

The property $\text{contains}(\text{namespace} : \text{Package}, \text{elements} : \text{Classifier})$ represents that a package contains classifiers. The interpretation $\mathcal{I}$ has the following interpretation of this property:
$\text{contains}^{\mathcal{I}} = \{(p1, c1), (p1, c2)(p1, pdt1), (p1, pdt2)\}$, such that there is only one package object $p1$, and it contains two classes ($c1$ and $c2$) and two primitive datatype objects ($pdt1$ and $pdt2$). This interpretation evaluates $(\text{namespace} \bullet \text{contains})^{\mathcal{I}}$ as the set $\{\{(p1, c1), (p1, c2), (p1, pdt1), (p1, pdt2)\}\}$ since there is only one object with role namespace which is the package object $p1$, and those elements in the opposite side of the property are those in $\text{contains}^{\mathcal{I}}$.

Given a Σ-formula $\varphi = (Obj, Rel, \Phi)$, a Σ-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$, and a valuation β, we can reduce an interpretation with respect to a formula by defining an interpretation $\mathcal{I}|_{(\varphi,\beta)} = (\boldsymbol{V_C^T}(\boldsymbol{O'}), \boldsymbol{A'})$ such that elements and properties in such interpretation are those with a corresponding element in the formula, i.e.

- $\boldsymbol{V_C^T}(\boldsymbol{O'}) = (V_c)_{c \in T(C)} \subseteq \boldsymbol{V_C^T}(\boldsymbol{O})$ with $\bigcup_c (V_c)_{c \in T(C)} = \beta(Obj)$
- $\boldsymbol{A'} = \{p^{\mathcal{I}}(\beta^c(x), \beta^d(y)) \in \boldsymbol{A} \mid \texttt{rel}(p, x^c, y^d) \in Rel\}$

In other words, since a model can be 'bigger' than the SW-model represented within the formula, it is reduced to coincide with the interpretation of the formula. We need this reduced model to prove whether the formula conforms to the metamodel. This is because we will only use the information within the model to state whether the constraints hold or not. In this sense, if the model provides more elements than those used for interpreting the SW-model represented within the formula, we can have the case in which the model satisfies the formula, but the SW-model does not conform with the metamodel. As an example of this, consider a SW-model with two elements of classes `A` and `B` and a link between them, and a multiplicity constraint of cardinality 1-2 between `A` and `B`. We can see that the SW-model does not conform to the metamodel, since we have only one element of class `B` linked to the element of class `A`. However, if we consider a model with an interpretation of those elements, plus one object of class `B` with a relation to the object of class `A`, this model does conform to the metamodel.

Example

The reduction of the interpretation with respect to the formula is as follows. We filter some elements, e.g. class $c2$, since they do not have a syntactic representation within the formula.

- A $\boldsymbol{T}(\boldsymbol{C})$-object domain consisting of

$$V_{\text{Class}} = \{c1\}$$
$$V_{\text{PrimitiveDataType}} = \{pdt1\}$$
$$V_{\text{Classifier}} = V_{\text{Class}} \cup V_{\text{PrimitiveDataType}}$$
$$V_{\text{Package}} = \{p1\}$$
$$V_{\text{Attribute}} = \{a1\}$$
$$V_{\text{UMLModelElement}} = V_{\text{Classifier}} \cup V_{\text{Package}} \cup V_{\text{Attribute}}$$
$$V_{\text{String}} = \{Pac, Str, ID, Per, val, nul\}$$

- A set $\boldsymbol{A}$ consisting of relations:

$$\text{contains}^{\mathcal{I}} = \{(p1, c1), (p1, pdt1)\}$$
$$\text{name}^{\mathcal{I}} = \{(p1, Pac), (c1, ID), (a1, val), (pdt1, Str)\}$$
$$\text{kind}^{\mathcal{I}} = \{(c1, Per), (pdt1, nul), (p, nul), (a, nul)\}$$
$$\text{typeOf}^{\mathcal{I}} = \{(a1, pdt1)\}$$
$$\text{has}^{\mathcal{I}} = \{(c1, a1)\}$$

Given Σ-interpretations $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$ and $\mathcal{I}' = (\boldsymbol{V_C^T}(\boldsymbol{O})', \boldsymbol{A}')$, a Σ-homomorphism $h : \mathcal{I} \to \mathcal{I}'$ is a family of maps $(h_c)_{c \in T(C)}$ with $h_c : V_c \to V_c'$ such that

- $h_c(v) \in O_c'$ forall $v \in O_c$
- $h_c(v) \in V_c' \backslash O_c'$ forall $v \in V_c \backslash O_c$
- $(v_1, v_2) \in p^{\mathcal{I}}$ iff $(h_{c_1}(v_1), h_{c_2}(v_2)) \in p^{\mathcal{I}'}$ for any $v_i \in V_c$ (i=1,2) for any $p(r_1 : c_1, r_2 : c_2) \in P$.

As stated in [CK08], we can show that Σ-homomorphisms can be composed, and that the composition of Σ-homomorphisms is associative. There also exists identity Σ-homomorphisms. Thus, Σ-interpretations and Σ-homomorphisms define a category (see Appendix A).

4.3 Satisfaction Relation

Given a signature Σ, a multiplicity constraint φ, and a Σ-interpretation $\mathcal{I}$, the interpretation satisfies φ, written $\mathcal{I} \models \varphi$, if one of the following conditions holds:

- φ is $\#(r \bullet p) = n$ and $|S| = n$ for all $S \in (r \bullet p)^{\mathcal{I}}$
- φ is $n \leq \#(r \bullet p)$ and $n \leq |S|$ for all $S \in (r \bullet p)^{\mathcal{I}}$
- φ is $\#(r \bullet p) \leq n$ and $|S| \leq n$ for all $S \in (r \bullet p)^{\mathcal{I}}$

This means that the number of elements related through a property p with any element with role r in such property, satisfy the multiplicity constraints.

Finally, given a signature Σ, a Σ-formula φ and a Σ-interpretation $\mathcal{I}$ we need to state the satisfaction relation. As the reader can notice, SW-models are well-typed by construction but may not satisfy the constraints. In this sense, the interpretation satisfies the formula if: (a) the SW-model has a correspondence with the interpretation $\mathcal{I}$ (isomorphic with respect to the reduced interpretation), and (b) the reduced interpretation satisfies the multiplicity constraints. Formally, given a signature $\Sigma = (\boldsymbol{T}, \boldsymbol{P})$ with $\boldsymbol{T} = (T(C), \leq_{T(C)}, C\,|_{abstract})$ and $\boldsymbol{P} = (R, P)$, a Σ-formula $\varphi = (Obj, Rel, \Phi)$ with variables $X = (X^c)_{c \in T(C)}$, a *valuation* $\beta = (\beta^c)_{c \in T(C)}$, and a Σ-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$, the *satisfaction relation* $\mathcal{I}, \beta \models_{\Sigma} \varphi$ holds if:

– $\forall\ \texttt{rel}(p, x^c, y^d) \in Rel, \exists\ p^{\mathcal{I}}(\beta^c(x), \beta^d(y)) \in \boldsymbol{A}$
– $\forall\ \varphi \in \Phi.\ \mathcal{I}|_{(\varphi,\beta)} \models \varphi$

Example

Now, we can check that $\mathcal{I}, \beta \models_{\Sigma} \varphi$ since:

– for all $\texttt{rel}(p, x^c, y^d) \in Rel$, there exists $p^{\mathcal{I}}(\beta^c(x), \beta^d(y)) \in \boldsymbol{A}$

$$\begin{gathered}
\text{for rel(contains, p, c)},\ (p1, c1) = (\beta(\text{p}), \beta(\text{c})) \in\ \text{contains}^{\mathcal{I}} \\
\text{for rel(contains, p, pdt)},\ (p1, pdt1) = (\beta(\text{p}), \beta(\text{pdt})) \in\ \text{contains}^{\mathcal{I}} \\
\text{for rel(has, c, a)},\ (c1, a1) = (\beta(\text{c}), \beta(\text{a})) \in\ \text{has}^{\mathcal{I}} \\
\text{for rel(typeOf, a, pdt)},\ (a1, pdt1) = (\beta(\text{a}), \beta(\text{pdt})) \in\ \text{typeOf}^{\mathcal{I}} \\
\text{for rel(name, p, Package)},\ (p1, Pac) = (\beta(\text{p}), \beta(\text{Package})) \in\ \text{name}^{\mathcal{I}} \\
\text{for rel(name, c, ID)},\ (c1, ID) = (\beta(\text{c}), \beta(\text{ID})) \in\ \text{name}^{\mathcal{I}} \\
\text{for rel(kind, c, Persistent)},\ (c1, Per) = (\beta(\text{c}), \beta(\text{Persistent})) \in\ \text{kind}^{\mathcal{I}} \\
\text{for rel(name, a, value), exists}\ (a1, val) = (\beta(\text{a}), \beta(\text{value})) \in\ \text{name}^{\mathcal{I}} \\
\text{for rel(name, pdt, String), exists}\ (pdt1, Str) = (\beta(\text{pdt}), \beta(\text{String})) \in\ \text{name}^{\mathcal{I}}
\end{gathered}$$

– $\mathcal{I}|_{(\varphi,\beta)} \models \varphi$ for all $\varphi \in \Phi$
  - $\#(\text{UMLModelElement} \bullet \text{name}) = 1$ and $|S| = 1$
    for all $S \in (\text{UMLModelElement} \bullet \text{name})^{\mathcal{I}}$ = $\{\{(p1, Pac)\}, \{(c1, ID)\}, \{(a1, val)\}, \{(pdt1, Str)\}\}$
  - $\#(\text{UMLModelElement} \bullet \text{kind}) = 1$ and $|S| = 1$
    for all $S \in (\text{UMLModelElement} \bullet \text{kind})^{\mathcal{I}}$ = $\{\{(c1, Per)\}, \{(pdt1, nul)\}, \{(a, nul)\}, \{(p, nul)\}\}$
  - $\#(\text{elements} \bullet \text{contains}) = 1$ and $|S| = 1$
    for all $S \in (\text{elements} \bullet \text{contains})^{\mathcal{I}}$ = $\{\{(p1, c1)\}, \{(p1, pdt1)\}\}$
  - $\#(\text{attribute} \bullet \text{has}) = 1$ and $|S| = 1$
    for all $S \in (\text{attribute} \bullet \text{has})^{\mathcal{I}}$ = $\{\{(c1, a1)\}\}$
  - $\#(\text{typeOpposite} \bullet \text{typeOf}) = 1$ and $|S| = 1$
    for all $S \in (\text{typeOpposite} \bullet \text{typeOf})^{\mathcal{I}}$ = $\{\{(a1, pdt1)\}\}$

Given signatures $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i)$ $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, and a Σ_2-interpretation $\mathcal{I} = (\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}), \boldsymbol{A})$, the reduct $\mathcal{I}|_\sigma$ of $\mathcal{I}$ along σ is the Σ_1-interpretation $\mathcal{I}_1 = (\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}|_\sigma), \boldsymbol{A}|_\sigma)$ with

– $\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}|_\sigma) = (V_{\sigma(c)})_{c \in T(C_1)}$
– $\boldsymbol{A}|_\sigma = \{\sigma_p(p)^{\mathcal{I}} \mid p \in P_1\}$

Moreover, given Σ_2-interpretations $\mathcal{I}_2 = (\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}_2), \boldsymbol{A}_2)$ and $\mathcal{I}'_2 = (\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}_2)', \boldsymbol{A}'_2)$, $\mathcal{I}_1$ denoting $\mathcal{I}_2|_\sigma$ and $\mathcal{I}'_1$ denoting $\mathcal{I}'_2|_\sigma$, and a Σ_2-homomorphism $h_2 : \mathcal{I}_2 \to \mathcal{I}'_2$, the reduct $h_2|_\sigma$ of h_2 along σ is the Σ_1-homomorphism $h_1 : \mathcal{I}_1 \to \mathcal{I}'_1$ defined by $h_{1_c}(v) = h_{2_{\sigma(c)}}(v)$ for any $c \in T(C_1)$, for any $v \in V_c$. It is possible to check that h_1 is indeed a Σ_1-homomorphism, since h_2 is a homomorphism and $h_2|_\sigma$ is defined for elements in $T(C_1)$.

As stated in [CK08], we can show that the reduct defines a functor (Lemma 4), and thus there is a functor Mod giving a category of interpretations for each signature and a functor defined by the reduct (Lemma 5).

Given variables $X = (X^c)_{c \in T(C)}$ and given a valuation β for X in I, the reduct of β along σ is a valuation for $X|_\sigma$ (as defined before) in $I|_\sigma$ defined by $(\beta|_\sigma)^c(x^c) \stackrel{\text{def}}{=} \beta^{\sigma(c)}(x^{\sigma(c)})$.

Finally, given signatures Σ_i, a signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$, a Σ_2-interpretation $\mathcal{I}$, a Σ_2-valuation β, and a Σ_1-formula ψ, the following satisfaction condition holds (see Appendix A).

$$\mathcal{I}|_\sigma, \beta|_\sigma \models_{\Sigma_1} \psi \text{ iff } \mathcal{I}, \beta \models_{\Sigma_2} \sigma(\psi)$$

Given that the satisfaction condition holds we can state that $\mathcal{I}^{\mathrm{C}}$ consisting of signatures, morphisms, formulas, interpretations, reducts, and the satisfaction relation, defines an institution.

5 An Institution for QVT-Relations

We finally introduce an institution $\mathcal{I}^{\mathrm{QVT}}$ for QVT-Relations check-only unidirectional transformations, and then we continue illustrating the concepts introduced with the example presented in Section 2.

5.1 Signatures and Formulas

A signature in $\mathcal{I}^{\mathrm{QVT}}$ is a triple $\langle \Sigma_1^{\mathrm{C}}, \Sigma_2^{\mathrm{C}}, \Sigma^{\mathrm{FOL}} \rangle$ with $\mathcal{I}^{\mathrm{C}}$-signatures $\Sigma_i^{\mathrm{C}} = (\boldsymbol{T}_i, \boldsymbol{P}_i)$ $(i = 1, 2)$ such that $\boldsymbol{T}_i$ = $(T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$ and $\boldsymbol{P}_i$ = (R_i, P_i), representing the source and target metamodels of the transformation, and a $FOL^{=}$ signature Σ^{FOL} such that there are sorts for every type ($\bigcup_i T(C_i) \subseteq S$) and there is a predicate for each property declaration ($\bigcup_i P_i \subseteq \Pi$). We assume that there are no name clashes (types, roles and properties) between source and target metamodels. In fact, if a transformation has the same source and target metamodels, we can use a prefix to identify elements on each side. A signature morphism is defined as a triple of morphisms of the corresponding institutions $\langle \sigma_1^{\mathrm{C}}, \sigma_2^{\mathrm{C}}, \sigma^{\mathrm{FOL}} \rangle$.

Example

The signature $\Sigma = \langle \Sigma_1^{\mathrm{C}}, \Sigma_2^{\mathrm{C}}, \Sigma^{\mathrm{FOL}} \rangle$ contains the signature Σ_1^{C} of the source metamodel, which is the one presented in the last subsection, the signature Σ_2^{C} of the target metamodel, which is not shown here but can be derived in the same way, and a $FOL^{=}$ signature Σ^{FOL} with at least one sort for each type name in $\bigcup_i T(C_i)$ and a predicate for each property in $\bigcup_i P_i$.

A Σ-formula is of the form $\langle \varphi_1^{\mathrm{C}}, \varphi_2^{\mathrm{C}}, \varphi^{\mathrm{rules}} \rangle$ such that φ_i^{C} is a Σ_i^{C}-formula and φ^{rules} is a formula representing the transformation specification.

Given a signature as defined before, and variables $X^s = (X^s)_{s \in S}$, a formula φ^{rules} is a tuple $\langle \mathrm{Rules}, \mathrm{top} \rangle$ such that Rules is the set of transformation rules, and $\mathrm{top} \subseteq \mathrm{Rules}$ the set of top rules of the transformation.

A rule $\mathrm{Rule} \in \mathrm{Rules}$ is a tuple of the form $\langle \mathrm{VarSet}, \mathrm{Pattern}_i \ (i = 1, 2), \mathrm{when}, \mathrm{where} \rangle$ such that $\mathrm{VarSet} \subseteq X^s$ is the set of variables of the rule, $\mathrm{Pattern}_i$ $(i = 1, 2)$ are the source and target patterns, and when/where are the `when`/`where` clauses of the rule, respectively. We will denote by k_VarSet the variables used in pattern k that do neither occur in the other domain nor in the `when` clause.

A pattern $\mathrm{Pattern}_i$ $(i = 1, 2)$ is a tuple $\langle E_i, A_i, Pr_i \rangle$ such that $E_i \subseteq (X^c)_{c \in C_i}$ is set of class-indexed variables, A_i is a set of elements representing associations of the form $rel(p, x, y)$ with $p \in P_i$ and $x, y \in E_i$, and Pr_i is a $FOL^{=}$-formula.

A `when` clause is a pair $\langle \mathrm{when}_c, \mathrm{when}_r \rangle$ such that when_c is a $FOL^{=}$-formula with variables in VarSet, and when_r is a set of pairs of transformation rules and set of variables which are the parameters used for the invocation of each rule. We will denote by WhenVarSet the set of variables ocurring in the `when` clause. Finally, a `where` clause is a pair $\langle \mathrm{where}_c, \mathrm{where}_r \rangle$ such that where_c is a $FOL^{=}$-formula with variables in VarSet, and where_r is a set of pairs of transformation rules and set of variables (parameters used for the invocation of each rule). Only variables used in a `where` clause (as `prefix` in the example) are contained in 2_VarSet.

Given a set of variables $X_2 = (X_2^s)_{s \in S_2}$, we define a set $X_2|_\sigma = X_1 = (X_1^{s_1})_{s_1 \in S_1}$ by $X_1^{s_1} = X_2^{\sigma(s_1)}$. Signature morphisms extend to formulas over Σ_1 and $X_2|_\sigma$ as follows. Given a Σ_1-formula $\varphi = \langle \varphi_1^{\mathrm{C}}, \varphi_2^{\mathrm{C}}, \varphi^{\mathrm{rules}} \rangle$, $\sigma(\varphi)$ is the canonical application of the signature morphism to every element in φ such that $\sigma(\varphi^{\mathrm{rules}})$ is the componentwise application of the $FOL^{=}$ and $\mathcal{I}^{\mathrm{C}}$ formula morphisms to every element in the rule.

We can show that the composition of signature morphisms is a signature morphism, that composition is associative, and identities are signature morphisms. Thus, signatures and signature morphisms define a category. Moreover, there is a functor Sen giving a set of formulas for each signature and a function translating sentences for each signature morphism. Proofs of these results can be found in Appendix B.

Example

A transformation between two SW-models is represented as a formula φ of the form $\langle \varphi_1^{\mathrm{C}}, \varphi_2^{\mathrm{C}}, \varphi^{\mathrm{rules}} \rangle$ such that, for example, φ_1^{C} is the formula introduced in the last subsection, which represents the source SW-model in Figure 3, φ_2^{C} is another formula representing the target SW-model (not shown here), and $\varphi^{\mathrm{rules}} = \langle \mathrm{Rules}, \mathrm{top} \rangle$ is the formula representing the transformation specification which is defined next.

The formula has three relations named `PackageToSchema` $\in$ top, `ClassToTable` $\in$ top and `AttributeToColumn`.

`PackageToSchema` = $\langle \mathrm{VarSet}, \mathrm{Pattern}_i\ (i = 1, 2), \mathrm{when}, \mathrm{where} \rangle$ such that

- $\mathrm{VarSet} = \{\mathrm{pn}, \mathrm{p}, \mathrm{s}\}$ with $\mathrm{pn} \in X^{String}$, $\mathrm{p} \in X^{Package}$, and $\mathrm{s} \in X^{Schema}$.
- $\mathrm{Pattern}_1 = \langle E_1, A_1, Pr_1 \rangle$ with $E_1 = \{\mathrm{p}\}$, $A_1 = \emptyset$, and $Pr_1 =$ `Class::name(p,pn)`. Remember that $\mathrm{name}(\mathrm{p}, \mathrm{pn})$ is a property in the source metamodel, and thus a predicate in the $FOL^{=}$ signature. Also observe that we use the class name as a prefix in order to avoid name clashing with those names in the target metamodel.
- $\mathrm{Pattern}_2 = \langle E_2, A_2, Pr_2 \rangle$ with $E_2 = \{\mathrm{s}\}$, $A_2 = \emptyset$, and $Pr_2 =$ `Relational::name(s,pn)`.
- $\mathrm{when} = \langle \emptyset, \emptyset \rangle$.
- $\mathrm{where} = \langle \emptyset, \emptyset \rangle$.

`ClassToTable` = $\langle \mathrm{VarSet}, \mathrm{Pattern}_i\ (i = 1, 2), \mathrm{when}, \mathrm{where} \rangle$ such that

- $\mathrm{VarSet} = \{\mathrm{cn}, \mathrm{prefix}, \mathrm{c}, \mathrm{p}, \mathrm{Persistent}, \mathrm{t}, \mathrm{s}, \mathrm{cl}, \mathrm{NUMBER}, \mathrm{cn} +' \_\mathrm{tid}', \mathrm{k}, \mathrm{cn} +' \_\mathrm{pk}'\}$ with $\mathrm{c} \in X^{Class}$, $\mathrm{p} \in X^{Package}$, $\mathrm{t} \in X^{Table}$, $\mathrm{s} \in X^{Schema}$, $\mathrm{cl} \in X^{Column}$, $\mathrm{k} \in X^{Key}$, and the others $\in X^{String}$.
- $\mathrm{Pattern}_1 = \langle E_1, A_1, Pr_1 \rangle$ with $E_1 = \{\mathrm{c}, \mathrm{p}\}$, $A_1 = \{rel(\mathrm{Class} :: \mathrm{contains}, \mathrm{p}, \mathrm{c})\}$, and $Pr_1 =$ `Class::name(c,cn) AND Class::kind(c,Persistent)`.
- $\mathrm{Pattern}_2 = \langle E_2, A_2, Pr_2 \rangle$ with $E_2 = \{\mathrm{t}, \mathrm{s}, \mathrm{cl}, \mathrm{k}\}$, $A_2 = \{rel(\mathrm{Relational} :: \mathrm{contains}, \mathrm{s}, \mathrm{t}), rel(\mathrm{Relational} :: \mathrm{has}, \mathrm{t}, \mathrm{cl}), rel(\mathrm{Relational} :: \mathrm{is}, \mathrm{cl}, \mathrm{k}), rel(\mathrm{Relational} :: \mathrm{has}, \mathrm{t}, \mathrm{k})\}$, and $Pr_2 =$

```
Relational::name(t,cn) AND
   Relational::name(cl,cn+'\_tid') AND
      Relational::type(cl,NUMBER) AND
        Relational::name(k,cn+'\_pk')
```

  .
- $\mathrm{when} = \langle \emptyset, \{(\mathtt{PackageToSchema}, \{p, s\})\} \rangle$.
- $\mathrm{where} = \langle \emptyset, \{(\mathtt{AttributeToColumn}, \{c, t, prefix\})\} \rangle$.

Example

`AttributeToColumn` = $\langle$VarSet, Pattern$_i$ $(i = 1, 2)$, when, where$\rangle$ such that

- VarSet = {an, pn, cn, sqltype, c, a, p, t, cl, prefix, EMPTY, INTEGER, NUMBER, BOOLEAN, VARCHAR, prefix $+'\_'+$ an} with an, pn, cn, sqltype, prefix, EMPTY, INTEGER, NUMBER, BOOLEAN, VARCHAR, prefix $+'\_'+$ an $\in X^{String}$, c $\in X^{Class}$, a $\in X^{Attribute}$, p $\in X^{PrimitiveDataType}$, t $\in X^{Table}$, and cl $\in X^{Column}$.
- Pattern$_1$ $= \langle E_1, A_1, Pr_1\rangle$ with E_1 = {c, a, p}, $A_1 = \{rel(\text{Class} :: \text{has}, \text{c}, \text{a}), rel(\text{Class} :: \text{typeOf}, \text{a}, \text{p})\}$, and Pr_1 = `Attribute::name(a,an) AND Class::name(p,pn).`
- Pattern$_2$ $= \langle E_2, A_2, Pr_2\rangle$ with E_2 = {t, cl}, $A_2 = \{rel(\text{Relational} :: \text{has}, \text{t}, \text{cl})\}$, and Pr_2 = `Relational::name(cl,cn) AND Relational::type(cl,sqltype)`
- when $= \langle\emptyset, \emptyset\rangle$.
- where $= \langle \text{where}_\text{c}, \emptyset\rangle$ with where$_\text{c}$ =

```
(((prefix = EMPTY) AND (cn = an)) OR
    (not (prefix = EMPTY) AND (cn = prefix+'\_'+an))) AND
      (((pn = INTEGER) AND (sqltype = NUMBER)) OR
    ((pn = BOOLEAN) AND (sqltype = BOOLEAN)) OR
       (((not (pn = INTEGER) AND (not (pn = BOOLEAN))
          AND (sqltype = VARCHAR)))
```

5.2 Models

A Σ-model is a triple $\langle \mathcal{M}_1^{\text{C}}, \mathcal{M}_2^{\text{C}}, \mathcal{M}^{\text{FOL}}\rangle$ of Sign$_i^{\text{C}}$ $(i = 1, 2)$ models, and a Sign$^{\text{FOL}}$ first-order structure, such that the interpretation of elements in Sign$_i^{\text{C}}$ must be the same in $\mathcal{M}_i^{\text{C}}$ and $\mathcal{M}^{\text{FOL}}$. This means that $|D|_t = V_t$. $\forall t \in \bigcup_i T(C_i)$, and $p_D = p^{\mathcal{I}}$. $\forall p \in \bigcup_i P_i$. In the case of $t \in T(C)\backslash C$ (primitive types) we have that $V_t \subseteq |D|_t$ since $\mathcal{M}^{\text{FOL}}$ can have more elements than those in the source and target institutions, as type constants (e.g. the empty string) and elements created using type constructors from other elements (e.g. new strings using type constructor $++$).

Given variables $X^s = (X^s)_{s \in S}$, the binding of a variable $x^c \in X^c$, denoted by $|x^c|$, is the set of any possible interpretation of such variable which corresponds to the carrier set of the corresponding sort, i.e. $|x^c| = |D|_c$. Moreover, the binding of a set of variables $(x_1, ..., x_n)$, denoted by $|(x_1, ..., x_n)|$, is defined as $\{(y_1, ..., y_n) \mid y_i \in |x_i|\ (i = 1..n)\}$. We can also view $|(x_1, ..., x_n)|$ as a set of variable assignments. We denote by $\mu[x_1, ..., x_n]$ the function with an assignment for variables $x_1, ..., x_n$. We also denote by $\mu_1 \cup \mu_2$ an assignment unifying the former ones, assuming that if there is variable clash, the assignment takes for those variables the values in μ_2.

Example

Binding of variables depends on the type of elements. If the variable is of a class, we have that the set of possible values coincides with the set of elements within the MOF institutions, i.e. $|D|_t = V_t$. For example, we have that |p| = V_{Package} = $\{p1\}$. However, if the variable is of a primitive type, we have than $V_t \subseteq |D|_t$ since transformation rules can use other elements besides those in the MOF institutions, for example those strings created using the type constructor $++$. In the example, we have that |pn| = $\{Pac, Str, ID, Per, val, nul, pk, tid, numb, varch, ... +tid, ID + +numb, ...\}$

Given signatures $\Sigma_i = \langle \Sigma_1^{\mathrm{C}}{}_i, \Sigma_2^{\mathrm{C}}{}_i, \Sigma^{\mathrm{FOL}}{}_i \rangle$ $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, and Σ_2-models $\mathcal{M} = \langle \mathcal{M}_1^{\mathrm{C}}, \mathcal{M}_2^{\mathrm{C}}, \mathcal{M}^{\mathrm{FOL}} \rangle$ and $\mathcal{M}_2 = \langle \mathcal{M}_1^{\mathrm{C}}{}_2, \mathcal{M}_2^{\mathrm{C}}{}_2, \mathcal{M}^{\mathrm{FOL}}{}_2 \rangle$, homomorphisms and reducts are defined componentwise. A Σ_2-homomorphism $h : \mathcal{M} \to \mathcal{M}_2$ is defined as a triple of homomorphisms $\langle h_1^{\mathrm{C}}, h_2^{\mathrm{C}}, h^{\mathrm{FOL}} \rangle$ of the corresponding institutions. The reduct $\mathcal{M}|_\sigma$ of $\mathcal{M}$ along σ is the Σ_1-interpretation $\langle \mathcal{M}_1^{\mathrm{C}}|_\sigma, \mathcal{M}_2^{\mathrm{C}}|_\sigma, \mathcal{M}^{\mathrm{FOL}}|_\sigma \rangle$. Moreover, the reduct $h|_\sigma$ of h along σ is the Σ_1-homomorphism $\langle h_1^{\mathrm{C}}|_\sigma, h_2^{\mathrm{C}}|_\sigma, h^{\mathrm{FOL}}{}_\sigma \rangle$. Notice that not every triple of reducts/homomorphisms is valid, it must ensure the above property on models.

We can show that Σ-homomorphisms can be composed, and that the composition of Σ-homomorphisms is associative. There also exist identity Σ-homomorphisms. Thus, Σ-models and Σ-homomorphisms define a category. We can also show that the reduct defines a functor, and thus there is a functor Mod giving a category of models for each signature and a functor defined by the reduct. Proofs of these results can be found in Appendix B.

Example

Assume that we have a model $\mathcal{M} = \langle \mathcal{M}_1^{\mathrm{C}}, \mathcal{M}_2^{\mathrm{C}}, \mathcal{M}^{\mathrm{FOL}} \rangle$ such that $\mathcal{M}_1^{\mathrm{C}} = (\mathcal{I}, \beta)$ as defined in the last subsection, $\mathcal{M}_2^{\mathrm{C}} = (\mathcal{I}', \beta')$ is a model with direct correspondence with the SW-model in the right side of Figure 3, and $\mathcal{M}^{\mathrm{FOL}}$ is a first-order structure.

5.3 Satisfaction Relation

A `when` clause $\langle \mathrm{when}_c, \mathrm{when}_r \rangle$ is satisfied with respect to a first-order structure $\mathcal{M}^{\mathrm{FOL}}$ and a variable assignment μ, denoted by $\mathcal{M}^{\mathrm{FOL}}, \mu \models \langle \mathrm{when}_c, \mathrm{when}_r \rangle$ if

$$\mathcal{M}^{\mathrm{FOL}}, \mu \models_{\mathrm{FOL}} \mathrm{when}_c \ \wedge \ (\forall (r, v) \in \mathrm{when}_r .\ \mathcal{M}^{\mathrm{FOL}}, \mu[v] \models r)$$

such that $\models_{\mathrm{FOL}}$ is the satisfaction relation in $FOL^{=}$, and the later is the satisfaction of the parametric transformation rule r using the variable assignment $\mu[v]$ as a parameter. The satisfaction of a `where` clause is defined in the same way.

A pattern $\mathrm{Pattern} = \langle E, A, Pr \rangle$ is satisfied with respect to a first-order structure $\mathcal{M}^{\mathrm{FOL}}$ and a variable assignment μ (which must include a valuation for the elements in E), denoted by $\mathcal{M}^{\mathrm{FOL}}, \mu \models \mathrm{Pattern}$ if there is a matching subgraph and the predicate holds, i.e.

- $\forall\, rel(p, x, y) \in A.\ p_D(\mu(x), \mu(y)) \in \ \mathcal{M}^{\mathrm{FOL}}$
- $\mathcal{M}^{\mathrm{FOL}}, \mu \models_{\mathrm{FOL}} Pr$

such that $\models_{\mathrm{FOL}}$ is the satisfaction relation in $FOL^{=}$.

A rule $\mathrm{Rule} = \langle \mathrm{VarSet}, \mathrm{Pattern}_i\ (i = 1, 2), \mathrm{when}, \mathrm{where} \rangle$ is satisfied with respect to a first-order structure $\mathcal{M}^{\mathrm{FOL}}$ and a variable assignment μ, denoted by $\mathcal{M}^{\mathrm{FOL}}, \mu \models \mathrm{Rule}$ if

1. If $\mathrm{WhenVarSet} = \emptyset$

$$\begin{array}{l} \forall\, \mu^1[x_1, ..., x_n] \in \ |\mathrm{VarSet} \backslash \mathrm{2\_VarSet}|, \\ \quad (\ \mathcal{M}^{\mathrm{FOL}}, (\mu^1[x_1, ..., x_n] \cup \mu) \models \mathrm{Pattern}_1 \rightarrow \\ \quad\quad \exists\, \mu^2[y_1, ..., y_m] \in |\mathrm{2\_VarSet}|, \\ \quad\quad\quad (\ \mathcal{M}^{\mathrm{FOL}}, (\mu^1 \cup \mu^2 \cup \mu) \models \mathrm{Pattern}_2 \wedge \\ \quad\quad\quad\quad \mathcal{M}^{\mathrm{FOL}}, (\mu^1 \cup \mu^2 \cup \mu) \models \mathrm{where})) \end{array}$$

2. If WhenVarSet $\neq \emptyset$

$$\begin{array}{l}\forall\, \mu^w[z_1,...,z_o] \in\ |\text{WhenVarSet}|,\\ \quad (\ \mathcal{M}^{\text{FOL}}, (\mu^w[z_1,...,z_o] \cup \mu) \models \text{when} \rightarrow\\ \quad\quad \forall\, \mu^1[x_1,...,x_n] \in\ |\text{VarSet}\backslash(\text{WhenVarSet} \cup \text{2\_VarSet})|,\\ \quad\quad\quad (\ \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^w \cup \mu) \models \text{Pattern}_1 \rightarrow\\ \quad\quad\quad\quad \exists\, \mu^2[y_1,...,y_m] \in |\text{2\_VarSet}|,\\ \quad\quad\quad\quad\quad (\ \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu^w \cup \mu) \models \text{Pattern}_2 \wedge\\ \quad\quad\quad\quad\quad\quad \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu^w \cup \mu) \models \text{where})))\end{array}$$

As with the institution for the conformance relation, we can reduce the model with respect to a formula. Given a formula $\varphi = \langle \varphi_1^{\text{C}}, \varphi_2^{\text{C}}, \varphi^{\text{rules}} \rangle$, and a model $\mathcal{M} = \langle \mathcal{M}_1^{\text{C}}, \mathcal{M}_2^{\text{C}}, \mathcal{M}^{\text{FOL}} \rangle$, we can define a model $\mathcal{M}|_\varphi = \langle \mathcal{M}_1^{\text{C}}|_\varphi, \mathcal{M}_2^{\text{C}}|_\varphi, \mathcal{M}^{\text{FOL}}|_\varphi \rangle$ such that:

- $\mathcal{M}_i^{\text{C}}|_\varphi$ $(i = 1, 2)$ is the reduction of $\mathcal{M}_i^{\text{C}} = (\mathcal{I}_i, \beta_i)$ with respect to the formula φ_i^{C}, which is $(\mathcal{I}_i|_{(\varphi_i^{\text{C}}, \beta_i)}, \beta_i)$, as defined for the institution for the conformance relation.
- $\mathcal{M}^{\text{FOL}}|_\varphi$ is the structure such that $|D|_t = V_t$. $\forall t \in \bigcup_i T(C_i)$ with $V_t \in \bigcup_i \boldsymbol{V_C^T}(\boldsymbol{O}_i)$, and $p_D = p^{\mathcal{I}|_{(\varphi_i^{\text{C}}, \beta_i)}}$. $\forall p \in \bigcup_i P_i$, with $\mathcal{I}_i|_{(\varphi_i, \beta_i)} = (\boldsymbol{V_C^T}(\boldsymbol{O}_i), \boldsymbol{A}_i)$.

Example

The reduction $\mathcal{M}^{\text{FOL}}|_\varphi$ is the structure such that the carrier sets corresponding to the interpretation of type names, and predicates, are those in $\mathcal{I}_i|_{(\varphi_i, \beta_i)}$ $(i = 1, 2)$. This means that the model only has an interpretation for those elements in the source and target SW-models, which is the same interpretation in the reduced source and target models of the institutions.

The satisfaction relation is defined such that a model $\mathcal{M}$ satisfies φ, written $\mathcal{M} \models_\Sigma \varphi$, if $\mathcal{M}_i^{\text{C}} \models^{\text{C}}_{\Sigma_i^{\text{C}}} \varphi_i^{\text{C}}$ $(i = 1, 2)$ and $\mathcal{M}|_\varphi \models_\Sigma \varphi^{\text{rules}}$. In other words, a model satisfies a formula if the SW-models conform to the corresponding metamodels, and they fulfill the top transformation rules. In this case we reduce the model in order to only consider elements with a representation within the formula. The satisfaction relation $\mathcal{M} \models_\Sigma \varphi^{\text{rules}}$ is defined to hold if for all $\text{Rule}_i \in \text{top}$. $\mathcal{M}^{\text{FOL}}, \emptyset \models \text{Rule}_i$. We take $\emptyset$ as the empty variable assignment, since for rules it will be used only in the case of non top and explicit called rules.

Example

We have that $\mathcal{M} \models_\Sigma \varphi$, if $\mathcal{M}_i^{\text{C}} \models^{\text{C}}_{\Sigma_i^{\text{C}}} \varphi_i^{\text{C}}$ $(i = 1, 2)$ and $\mathcal{M}|_\varphi \models_\Sigma \varphi^{\text{rules}}$. We already showed that $\mathcal{M}_1^{\text{C}} \models^{\text{C}}_{\Sigma_1^{\text{C}}} \varphi_1^{\text{C}}$, and we can prove in the same way that $\mathcal{M}_2^{\text{C}} \models^{\text{C}}_{\Sigma_2^{\text{C}}} \varphi_2^{\text{C}}$. Thus, we need to prove that $\mathcal{M}|_\varphi \models_\Sigma \varphi^{\text{rules}}$, and this holds if $\mathcal{M}^{\text{FOL}}|_\varphi, \emptyset \models \texttt{ClassToTable}$, and $\mathcal{M}^{\text{FOL}}|_\varphi, \emptyset \models \texttt{PackageToSchema}$.

We first prove that $\mathcal{M}^{\text{FOL}}|_\varphi\ \emptyset \models \texttt{PackageToSchema}$. We know that $|\text{pn}| = \{Pac, Str, ID, Per, val, nul, pk, ti$ and $|\text{p}| = V_{\text{Package}} = \{p1\}$, so $|\{\text{pn}, \text{p}\}|$ is $\{(Pac, p1), (Str, p1), (ID, p1), (Per, p1), (val, p1), ...\}$. We also

have that |s| = $V_{\text{Schema}} = \{s1\}$. Thus, $\mathcal{M}^{\text{FOL}}|_\varphi, \emptyset \models$ `PackageToSchema` if

$$\begin{array}{l}
\forall\, \mu^1[\text{pn}, \text{p}] \in \{(Pac, p1), (Str, p1), (ID, p1), (Per, p1), (val, p1), (nul, p1), ...\}, \\
\quad (\, \mathcal{M}^{\text{FOL}}|_\varphi, \mu^1 \models \text{Pattern}_1 \rightarrow \\
\qquad \exists\, \mu^2[\text{s}] \in \{s1\}, \\
\qquad\quad (\, \mathcal{M}^{\text{FOL}}|_\varphi, (\mu^1 \cup \mu^2) \models \text{Pattern}_2 \wedge \\
\qquad\qquad \mathcal{M}^{\text{FOL}}|_\varphi, (\mu^1 \cup \mu^2) \models \text{where}))
\end{array}$$

For every $\mu^1[\text{pn}, \text{p}]$ different from $(Pac, p1)$ we have that Pattern_1 does not hold, since it depends on the predicate `Class::name(p,pn)`. Thus, in these cases the rest of the implication holds. Now, in the case of $(Pac, p1)$, we have that Pattern_1 holds, and that the only possible value for s is $s1$. In this case, we also have that $\mathcal{M}^{\text{FOL}}|_\varphi, (\mu^1 \cup \mu^2) \models \text{Pattern}_2$ since the predicate `Relational::name(s,pn)` holds. Note in the left side of Figure 3 that the schema has the same name as the package, which is semantically represented as Pac. Moreover, since the `where` clause is empty, $\mathcal{M}^{\text{FOL}}|_\varphi, (\mu^1 \cup \mu^2) \models \text{where}$ trivially holds. Finally, we conclude that $\mathcal{M}^{\text{FOL}}|_\varphi \models$ `PackageToSchema` indeed.

We now prove that $\mathcal{M}^{\text{FOL}}|_\varphi, \emptyset \models$ `ClassToTable`. Proceeding in the same way, we have to prove that:

$$\begin{array}{l}
\forall\, \mu^w[\text{p}, \text{s}] \in \{(p1, s1)\}, \\
\quad (\, \mathcal{M}^{\text{FOL}}, \mu^w[\text{p}, \text{s}] \models \text{when} \rightarrow \\
\qquad \forall\, \mu^1 \in |(\text{cn}, \text{c}, \text{Persistent})|, \\
\qquad\quad (\, \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^w) \models \text{Pattern}_1 \rightarrow \\
\qquad\qquad \exists\, \mu^2 \in |(\text{prefix}, \text{t}, \text{cl}, \text{NUMBER}, \text{cn} +' \text{_tid}', \text{k}, \text{cn} +' \text{_pk}')|, \\
\qquad\qquad\quad (\, \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu^w) \models \text{Pattern}_2 \wedge \\
\qquad\qquad\qquad \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu^w) \models \text{where})))
\end{array}$$

In this case we have a `when` clause which is the invocation of the relation `PackageToSchema` with a concrete variable assignment for domain variables p and s. We proved above that with this assignment $\mathcal{M}^{\text{FOL}}|_\varphi, \mu^w[\text{p}, \text{s}] \models$ `PackageToSchema` holds. For proving $\mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^w) \models \text{Pattern}_1$ we need to prove that $\text{Class} :: \text{contains}_D(\mu(\text{p}), \mu(\text{c})) \in \mathcal{M}^{\text{FOL}}$ since $rel(\text{Class} :: \text{contains}, \text{p}, \text{c}) \in A$, and also that $\mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^w) \models_{\text{FOL}}$ `Class::name(c,cn) AND Class::kind(c,Persistent)`. This only holds with the variable assignment $\mu^1[\text{c}, \text{cn}, \text{Persistent}] = (c1, ID, Per)$ and $\mu^w[\text{p}] = p1$. In any other case, Pattern_1 does not hold and thus the rest of the expression holds.

Now, for proving $\mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2) \models \text{Pattern}_2$ we need to prove that

- $\text{Relational} :: \text{contains}_D(\mu(\text{s}), \mu(\text{t})) \in \mathcal{M}^{\text{FOL}}$ since $rel(\text{Relational} :: \text{contains}, \text{s}, \text{t}) \in A$,
- $\text{Relational} :: \text{has}_D(\mu(\text{t}), \mu(\text{cl})) \in \mathcal{M}^{\text{FOL}}$ since $rel(\text{Relational} :: \text{has}, \text{t}, \text{cl}) \in A$,
- $\text{Relational} :: \text{is}_D(\mu(\text{cl}), \mu(\text{k})) \in \mathcal{M}^{\text{FOL}}$ since $rel(\text{Relational} :: \text{is}, \text{cl}, \text{k}) \in A$,
- $\text{Relational} :: \text{has}_D(\mu(\text{t}), \mu(\text{k})) \in \mathcal{M}^{\text{FOL}}$ since $rel(\text{Relational} :: \text{has}, \text{t}, \text{k}) \in A$,

and also that $\mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2) \models_{\text{FOL}} Pr_2$, with

$Pr_2 =$

```
Relational::name(t,cn) AND
   Relational::name(cl,cn+'\_tid') AND
      Relational::type(cl,NUMBER) AND
        Relational::name(k,cn+'\_pk')
```

These hold with $\mu^2[\text{prefix}, \text{t}, \text{cl}, \text{NUMBER}, \text{cn} +' \text{\_tid}', \text{k}, \text{cn} +' \text{\_pk}']$
= $(nul, t1, cl1, num, ID{+}{+}\_tid, k1, ID{+}{+}\_pk)$. Finally, with the variable assignment we have at the moment
$(\mu^1 \cup \mu^2 \cup \mu^w)[\text{cn}, \text{c}, \text{p}, \text{Persistent}, \text{prefix}, \text{t}, \text{s}, \text{cl}, \text{NUMBER}, \text{cn} +' \text{\_tid}', \text{k}, \text{cn} +' \text{\_pk}'] =$
$(ID, c1, p1, Per, nul, t1, s1, cl1, num, ID + +\_tid, k1, ID + +\_pk)$ we can prove
$\mathcal{M}^{\text{FOL}}, \mu[c, t, prefix] \models$ `AttributeToColumn`.

As before, we have to prove that:

$$\begin{aligned}
&\forall\, \mu^1[...] \in\ |\text{c}, \text{a}, \text{p}, \text{an}, \text{pn}|,\\
&\quad (\ \mathcal{M}^{\text{FOL}}, (\mu^1[\text{c}, \text{a}, \text{p}, \text{an}, \text{pn}] \cup \mu[\text{c}, \text{t}, \text{prefix}]) \models \text{Pattern}_1 \rightarrow\\
&\qquad \exists\, \mu^2[...] \in |\text{cn}, \text{sqltype}, \text{t}, \text{cl}, \text{prefix}, \text{EMPTY}, \text{INTEGER},\\
&\qquad\qquad \text{NUMBER}, \text{BOOLEAN}, \text{VARCHAR}, \text{prefix} +' \text{_}' + \text{an}|,\\
&\qquad\quad (\ \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu) \models \text{Pattern}_2 \wedge\\
&\qquad\qquad \mathcal{M}^{\text{FOL}}, (\mu^1 \cup \mu^2 \cup \mu) \models \text{where}))
\end{aligned}$$

For every $\mu^1[\text{c}, \text{a}, \text{p}, \text{an}, \text{pn}]$ different from $(c1, a1, pdt1, val, Str)$ we have that Pattern_1 does not hold, since it depends on the predicate
`Attribute::name(a,an) AND Class::name(p,pn)`. Thus, in these cases the rest of the expression holds. Now, in the case of $(c1, a1, pdt1, val, Str)$, we have that Pattern_1 holds. In this case, there exists a variable assignment
$\mu^2[\text{cn}, \text{sqltype}, \text{t}, \text{cl}, \text{prefix}, \text{EMPTY}, \text{INTEGER}, \text{NUMBER}, \text{BOOLEAN}, \text{VARCHAR}, \text{prefix} +' \text{\_}' + \text{an}] =$
$(val, varch, t1, cl2, nul, nul, int, numb, BOOL, VARC, NUL\_val)$ such that
$\mathcal{M}^{\text{FOL}}|_\varphi, (\mu^1 \cup \mu^2 \cup \mu) \models \text{Pattern}_2$. This can be viewed in the left side of Figure 3, where the table t (semantically represented as $t1$) has a column c2 (semantically represented as $cl2$) with column name value (semantically represented as val) and type VARCHAR (semantically represented as $VARC$) which satisfies the predicate
`Relational::name(cl,cn) AND Relational::type(cl,sqltype)`. Finally, the same variable assignment satisfies the `where` clause since
`(prefix = EMPTY) AND (cn = an)`, and also
`(not (pn = INTEGER) AND (not (pn = BOOLEAN))) AND (sqltype = VARCHAR).`

Finally, given signatures Σ_i, a signature morphism $\sigma : \Sigma_1 \rightarrow \Sigma_2$, a Σ_2-model $\mathcal{M}$, a set of variables X_2, and a Σ_1-formula ψ with variables in $X_2|_\sigma$, the following satisfaction condition holds (see Appendix B).

$$\mathcal{M}|_\sigma \models_{\Sigma_1} \psi \text{ iff } \mathcal{M} \models_{\Sigma_2} \sigma(\psi)$$

Given that the satisfaction condition holds we can state that $\mathcal{I}^{\text{QVT}}$ consisting of signatures, morphisms, formulas, interpretation, reducts, and the satisfaction relation, defines an institution.

6 Alternative Institutions for MOF and QVT

In this section we present an alternative approach to the formal definition of the institutions $\mathcal{I}^{\mathrm{C}}$ and $\mathcal{I}^{\mathrm{QVT}}$ for the conformance relation and QVT-Relations check-only unidirectional transformations, respectively. We basically change the definition of $\mathcal{I}^{\mathrm{C}}$ signatures in close relation with the ideas presented in [JKMR13] in which instances (class objects and type values) are represented within the signature instead of within the formulas, as defined in Figure 8. However, in [JKMR13] there is no representation of links within the signature. This change simplifies several definition, e.g avoids the need of reducing the interpretation with respect to the elements in the formula to prove the satisfaction relation.

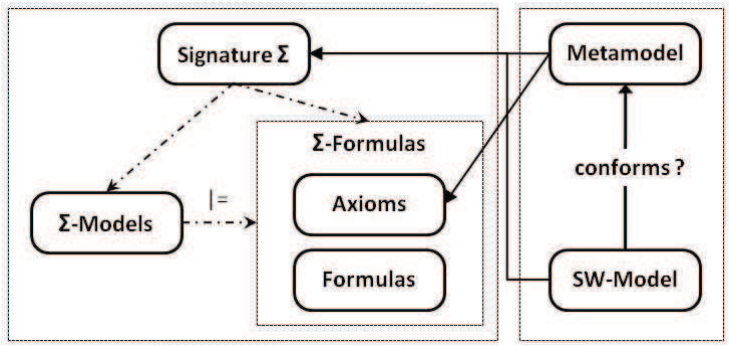


Fig. 8: An alternative approach to the conformance relation as an institution

Along the definition we will illustrate the concepts introduced with the example presented in Section 2.

6.1 An Alternative Institution for MOF

In what follows we only present those definitions that change from the former ones.
A signature $\Sigma = (\boldsymbol{T}, \boldsymbol{P}, \boldsymbol{M})$ declares:

- a type extension of a finite class hierarchy $\boldsymbol{T} = (T(C), \leq_{T(C)}, C|_{abstract})$ extended with a subset $C|_{abstract} \subseteq C$ denoting abstract classes;
- a properties declaration (attributes and associations) $\boldsymbol{P} = (R, P)$ where R is a finite set of role names and P is a finite set $(p_w)_{w \in (R \times T(C)) \times (R \times T(C))}$ of property names indexed over pairs of a role name and a class (or type) name, such that for any class or type name $c \in C$, the role names of the properties in which any $c' \leq_{T(C)} c$ is involved are all different; and
- a SW-model declaration (instances and links) $\boldsymbol{M} = (I, L)$ where I is a finite set of instances of the form $o : c$ with $c \in T(C)$; and L is a finite set of links between instances of the form $p_w(x, y)$ with $p_w \in P$, $w = ((r_1, c)(r_2, d))$, $x : c,\ y : d\ \in I$.

Example

From the class metamodel in Figure 2 and the SW-model in Figure 3 we can derive the signature $(\boldsymbol{T}, \boldsymbol{P}, \boldsymbol{M})$ such that:

$$T(C) = \{\text{UMLModelElement}, \text{Package}, ..., \text{String}\}$$
$$\leq_{T(C)} = \{\text{Package} \leq_{T(C)} \text{UMLModelElement}, ...\}$$
$$C|_{abstract} = \{\text{UMLModelElement}\}$$

$\boldsymbol{P} = (R, P)$ where

$$R = \{\text{namespace}, \text{elements}, \text{type}, \text{typeOpposite}, ...\}$$

$$P = \{\text{contains}(\text{namespace} : \text{Package}, \text{elements} : \text{Classifier}),$$
$$\text{name}(\text{UMLModelElement} : \text{UMLModelElement}, \text{name} : \text{String}),$$
$$\text{typeOf}(\text{typeOpposite} : \text{Attribute}, \text{type} : \text{PrimitiveDataType}), ...\}$$

$$I = \{\text{p} : \text{Package}, \text{c} : \text{Class}, \text{a} : \text{Attribute}, ..., \text{String} : \text{String}, \text{NULL} : \text{String}\}$$
$$L = \{\text{contains}(\text{p}, \text{c}), \text{contains}(\text{p}, \text{pdt}), \text{has}(\text{c}, \text{a}), \text{type}(\text{a}, \text{pdt}), \text{name}(\text{p}, \text{Package}),$$
$$\text{kind}(\text{c}, \text{Persistent}), \text{name}(\text{pdt}, \text{String}), ...\}$$

Since the SW-model is now represented within the signature, the formulas only represent multiplicity constraints. Thus, given a signature as defined before, any Σ-formula is defined by:

$$\Phi ::= \#\Pi = n \mid n \leq \#\Pi \mid \#\Pi \leq n$$
$$\Pi ::= R \bullet P$$

where $n \in \mathbb{N}$. The $\#$-expressions return the number of links in a property when some roles are fixed. For this, we use $\bullet$ as the select/partition operator in Π representing the selection of the elements in the opposite side of role R in property P.

Example

The set of formulas φ corresponding to the metamodel in Figure 2 is defined by:

$$\varphi = \{\#(\text{UMLModelElement} \bullet \text{name}) = 1, \#(\text{UMLModelElement} \bullet \text{kind}) = 1,$$
$$\#(\text{elements} \bullet \text{contains}) = 1, \#(\text{attribute} \bullet \text{has}) = 1, ...\}.$$

Let $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i, \boldsymbol{M}_i)$ $(i = 1, 2)$ with $\boldsymbol{T}_i = (T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$, $\boldsymbol{P}_i = (R_i, P_i)$, and $\boldsymbol{M}_i = (I_i, L_i)$. A signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ is a tuple of maps $\langle \sigma_T, \sigma_R, \sigma_P, \sigma_I \rangle$ between class names, role names, property names, and instances, such that the following conditions hold:

- $a \in C_1$ implies $\sigma_T(a) \in C_2$,
- $a \in T(C_1) \backslash C_1$ implies $\sigma_T(a) \in T(C_2) \backslash C_2$,
- $a, b \in T(C_1)$ with $a \leq_{T(C_1)} b$ implies $\sigma_T(a) \leq_{T(C_2)} \sigma_T(b)$,
- $a \in C_1|_{abstract}$ implies $\sigma_T(a) \in C_2|_{abstract}$,
- $p_w \in P_1$ implies $\sigma_P(p)_{\sigma(w)} \in P_2$, where σ is the canonical extension of σ_T and σ_R to words in $(R \times T(C))(R \times T(C))$.
- $o : c \in I_1$ implies $\sigma_I(o) : \sigma_T(c) \in I_2$
- $p_w(x, y) \in L_1$ implies $\sigma_P(p)_{\sigma(w)}(\sigma_I(x), \sigma_I(y)) \in L_2$

Signature morphisms extend to formulas over Σ_1 as follows. Given a Σ_1-formula φ, $\sigma(\varphi)$ is the canonical application of the signature morphism to every role and property in the formula such that $\sigma(r \bullet p) = \sigma_R(r) \bullet \sigma_P(p)$.

We adapt the definition of a Σ-interpretation in order to 'reduce' the interpretation to those elements and relations in $\boldsymbol{M}$, i.e. there is an isomorphism between these elements and those in the interpretation. A Σ-interpretation $\mathcal{I}$ consists of a tuple $(\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A}, K^{\mathcal{I}})$ where

- $\boldsymbol{V_C^T}(\boldsymbol{O}) = (V_c)_{c \in T(C)}$ is a $\boldsymbol{T}(\boldsymbol{C})$-object domain
- $\boldsymbol{A}$ contains a relation $p^{\mathcal{I}} \subseteq V_{c_1} \times V_{c_2}$ for each relation name $p(r_1 : c_1, r_2 : c_2) \in P$ with $c_1, c_2 \in T(C)$
- $K^{\mathcal{I}}$ maps each $o : c \in I$ to an element of V_c
- $c_2 \in C|_{abstract}$ implies $O_{c_2} = \bigcup_{c_1 \leq_C c_2} O_{c_1}$
- $K^{\mathcal{I}}(o_1 : c) \neq K^{\mathcal{I}}(o_2 : d)$ iff $o_1 : c \neq o_2 : d$
- $V_c = \bigcup_c K^{\mathcal{I}}(o : c)$ with $o : c \in I$, for all $c \in T(C)$
- $p^{\mathcal{I}} = \{(K^{\mathcal{I}}(x : c), K^{\mathcal{I}}(y : d)) \mid p_w(x, y) \in L, x : c,\ y : d\ \in I\}$

Example

An interpretation $\mathcal{I}$ can be as follows, in which each element has a correspondence with one in the signature:

- A $\boldsymbol{T}(\boldsymbol{C})$-object domain consisting of

$$V_{\text{Class}} = \{c1\}$$
$$V_{\text{PrimitiveDataType}} = \{pdt1\}$$
$$V_{\text{Classifier}} = V_{\text{Class}} \cup V_{\text{PrimitiveDataType}}$$
$$V_{\text{Package}} = \{p1\}$$
$$V_{\text{Attribute}} = \{a1\}$$
$$V_{\text{UMLModelElement}} = V_{\text{Classifier}} \cup V_{\text{Package}} \cup V_{\text{Attribute}}$$
$$V_{\text{String}} = \{Pac, Str, Per, nul, ID, val\}$$

- A set $\boldsymbol{A}$ consisting of relations:

$$\text{contains}^{\mathcal{I}} = \{(p1, c1), (p1, pdt1)\}$$
$$\text{name}^{\mathcal{I}} = \{(p1, Pac), (c1, ID), (c2, nul), (a1, val)\}$$
$$\text{kind}^{\mathcal{I}} = \{(p1, nul), (c1, Per), (a1, nul), (pdt1, nul)\}$$
$$\text{type}^{\mathcal{I}} = \{(a1, pdt1)\}$$
$$\ldots$$

Given Σ-interpretations $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A}, K^{\mathcal{I}})$ and $\mathcal{I}' = (\boldsymbol{V_C^T}(\boldsymbol{O})', \boldsymbol{A}', K^{\mathcal{I}'})$, a Σ-homomorphism $h : \mathcal{I} \to \mathcal{I}'$ is a family of maps $(h_c)_{c \in T(C)}$ with $h_c : V_c \to V_c'$ such that

- $h_c(v) \in O_c'$ forall $v \in O_c$
- $h_c(v) \in V_c' \backslash O_c'$ forall $v \in V_c \backslash O_c$
- $(v_1, v_2) \in p^{\mathcal{I}}$ iff $(h_{c_1}(v_1), h_{c_2}(v_2)) \in p^{\mathcal{I}'}$ for any $v_i \in V_c$ (i=1,2), $p(r_1 : c_1, r_2 : c_2) \in P$.
- $h_c(K^{\mathcal{I}}(o_1 : c)) \neq h_c(K^{\mathcal{I}}(o_2 : d))$ iff $K^{\mathcal{I}}(o_1 : c) \neq K^{\mathcal{I}}(o_2 : d)$

Given a signature Σ, a formula φ, and a Σ-interpretation $\mathcal{I}$, the interpretation satisfies φ, written $\mathcal{I} \models_\Sigma \varphi$, if one of the following conditions holds:

- φ is $\#(r \bullet p) = n$ and $|S| = n$ for all $S \in (r \bullet p)^{\mathcal{I}}$
- φ is $n \leq \#(r \bullet p)$ and $n \leq |S|$ for all $S \in (r \bullet p)^{\mathcal{I}}$
- φ is $\#(r \bullet p) \leq n$ and $|S| \leq n$ for all $S \in (r \bullet p)^{\mathcal{I}}$

This means that the number of elements related through a property p with any element with role r in such property, satisfy the multiplicity constraints. This definition can be trivially extended for a set of formulas Φ, as follows: $\mathcal{I} \models_\Sigma \Phi$ iff $\mathcal{I} \models_\Sigma \varphi$. $\forall \varphi \in \Phi$.

Example

Now, we can check that $\mathcal{I}, \beta \models_\Sigma \varphi$ for every formula φ defined before.

- $\#(\text{UMLModelElement} \bullet \text{name}) = 1$ and $|S| = 1$
  for all $S \in (\text{UMLModelElement} \bullet \text{name})^\mathcal{I}$ = $\{\{(p1, Pac)\}, \{(c1, ID)\}, \{(a1, val)\}, \{(pdt1, Str)\}\}$

- $\#(\text{UMLModelElement} \bullet \text{kind}) = 1$ and $|S| = 1$
  for all $S \in (\text{UMLModelElement} \bullet \text{kind})^\mathcal{I}$ = $\{\{(c1, Per)\}, \{(pdt1, nul)\}, \{(a, nul)\}, \{(p, nul)\}\}$

- $\#(\text{elements} \bullet \text{contains}) = 1$ and $|S| = 1$
  for all $S \in (\text{elements} \bullet \text{contains})^\mathcal{I}$ = $\{\{(p1, c1)\}, \{(p1, pdt1)\}\}$

- $\#(\text{attribute} \bullet \text{has}) = 1$ and $|S| = 1$
  for all $S \in (\text{attribute} \bullet \text{has})^\mathcal{I}$ = $\{\{(c1, a1)\}\}$

- $\#(\text{typeOpposite} \bullet \text{typeOf}) = 1$ and $|S| = 1$
  for all $S \in (\text{typeOpposite} \bullet \text{typeOf})^\mathcal{I}$ = $\{\{(a1, pdt1)\}\}$

Given signatures $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i, \boldsymbol{M}_i)$ $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, and a Σ_2-interpretation $\mathcal{I}$ = $(\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}), \boldsymbol{A}, K^\mathcal{I})$, the reduct $\mathcal{I}|_\sigma$ of $\mathcal{I}$ along σ is the Σ_1-interpretation $\mathcal{I}_1$ = $(\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}|_\sigma), \boldsymbol{A}|_\sigma, K^\mathcal{I}|_\sigma)$ with

- $\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}|_\sigma)$ = $(V_{\sigma(c)})_{c \in T(C_1)}$
- $\boldsymbol{A}|_\sigma$ = $\{\sigma_p(p)^\mathcal{I} \mid p \in P_1\}$
- $K^\mathcal{I}|_\sigma$ $(o : c)$ = $K^\mathcal{I}(\sigma_I(o) : \sigma_T(c))$ for all $o : c \in I_1$

Finally, the satisfaction condition holds for given signatures Σ_i $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-interpretation $\mathcal{I}$, and a Σ_1-formula ψ (it can be trivially extended to a set of formulas):

$$\mathcal{I}|_\sigma, \models_{\Sigma_1} \psi \text{ iff } \mathcal{I} \models_{\Sigma_2} \sigma(\psi)$$

Categorical and functorial properties of the institution, as well as the satisfaction condition are proved in Appendix C. Given that the satisfaction condition holds we can state that $\mathcal{I}^{\mathrm{C}}$ consisting of signatures, morphisms, formulas, interpretation, reducts, and the satisfaction relation, defines an institution.

6.2 An Alternative Institution for QVT-Relations

We define a second institution $\mathcal{I}^{\mathrm{QVT}}$ for QVT-Relations check-only unidirectional transformations based on the alternative definition of the institution $\mathcal{I}^{\mathrm{C}}$ for the conformance relation.

The only difference here is that since the SW-model is represented within the signature, we do not need to reduce the model. This means that a model $\mathcal{M}$ satisfies φ, written $\mathcal{M} \models_\Sigma \varphi$, if $\mathcal{M}^{\mathrm{C}}_i \models^{\mathrm{C}}_{\Sigma^{\mathrm{C}}_i} \varphi^{\mathrm{C}}_i$ $(i = 1, 2)$ and $\mathcal{M} \models_\Sigma \varphi^{\mathrm{rules}}$.

Proof of the satisfaction condition is in Appendix C. Given that the satisfaction condition holds we can state that $\mathcal{I}^{\mathrm{QVT}}$ consisting of signatures, morphisms, formulas, interpretations, reducts, and the satisfaction relation, defines an institution.

7 Conclusions and Future Work

In this paper we have defined institutions to represent the conformance relation between MOF models and metamodels, and the satisfaction of QVT-Relations check-only unidirectional transformations between pairs of models. These definitions neither depend on a shallow embedding of the languages by providing a syntactic translation into other logics, nor on the definition of specific institutions for each metamodel or model transformation. On the contrary, we defined a generic and minimal infrastructure within a theory which allows the definition of semantic-preserving translations from the MDE elements to potentially any logic defined as an institution, with the advantage that there is no need of maintain multiple formal representations of the same MDE elements.

Unlike MOF, we do not consider derived relations, n-ary properties, aggregation, uniqueness and ordering properties within a property end, operations on classes, or packages, since they are elements not commonly used within transformations. We neither consider black-box operations or rule and transformation overriding within transformations since there are advanced features not commonly used in practice, nor keys definition since there are used for object creation not within the checking semantics. However, an inclusion of these elements within our institutions will strengthen the formal environment for MDE.

Our institutions contribute to the definition of a comprehensive formal environment for the verification of model transformations. We plan to define comorphisms from our institutions to a host logic and supplement this information with properties specified in the host logic. In particular, we are setting up an appropriate comorphism to CASL, the main language within Hets. This tool provides proof management capabilities for monitoring the overall correctness of a heterogeneous specification whereas different parts of it are verified using (possibly different) proof systems. Moreover, the graph of logics already defined within Hets, reduces the complexity of linking different semantic domains to perform a comprehensive verification using multiple semantic domains.

Our medium-term goals are the development of a first functional prototype and its validation. Although we are for now focusing on MOF and QVT-Relations, we envision in the long-term to extend the environment to support other transformation approaches.

References

ABK07. Kyriakos Anastasakis, Behzad Bordbar, and Jochen M. Küster. Analysis of model transformations via Alloy. In Benoit Baudry, Alain Faivre, Sudipto Ghosh, and Alexander Pretschner, editors, *Proceedings of the 4th MoDeVVa workshop Model-Driven Engineering, Verification and Validation*, pages 47–56, 2007.

BHM09. Artur Boronat, Reiko Heckel, and José Meseguer. Rewriting logic semantics and verification of model transformations. In *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering (FASE'09)*, pages 18–33. Springer, 2009.

BKMW08. Artur Boronat, Alexander Knapp, José Meseguer, and Martin Wirsing. What is a multi-modeling language? In Andrea Corradini and Ugo Montanari, editors, *WADT*, volume 5486 of *Lecture Notes in Computer Science*, pages 71–87. Springer, 2008.

BKS02. Bernhard Beckert, Uwe Keller, and Peter Schmitt. Translating the Object Constraint Language into first-order predicate logic. In *Proceedings of the VERIFY Workshop at Federated Logic Conferences (FLoC), Denmark*, 2002.

BM09. Artur Boronat and José Meseguer. Algebraic semantics of OCL-constrained metamodel specifications. In Manuel Oriol, Bertrand Meyer, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski, editors, *Objects, Components, Models and Patterns*, volume 33 of *Lecture Notes in Business Information Processing*, pages 96–115. Springer, 2009.

CH06. K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Systems Journal*, 45(3):621–645, 2006.

CK08. María Victoria Cengarle and Alexander Knapp. An institution for UML 2.0 static structures. Technical Report TUM-I0807, Institut für Informatik, Technische Universität München, 2008.

CKTW08. María Victoria Cengarle, Alexander Knapp, Andrzej Tarlecki, and Martin Wirsing. *A Heterogeneous Approach to UML Semantics*, volume 5065 of *Lecture Notes in Computer Science*, chapter chapter 23, pages 383–402. Springer Berlin Heidelberg, 2008.

CLST10. Daniel Calegari, Carlos Luna, Nora Szasz, and Alvaro Tasistro. Representation of metamodels using inductive types in a type-theoretic framework for MDE. Technical Report RT10-01, InCo-PEDECIBA, 2010.

CS13. Daniel Calegari and Nora Szasz. Bridging techological spaces for the verification of model transformations. In *Proc. Conf. Iberoamericana de Software Engineering (CIbSE)*, 2013.

Fav09. Liliana Favre. A formal foundation for metamodeling. In *Proceedings of the 14th Ada-Europe International Conference on Reliable Software Technologies*, Ada-Europe '09, pages 177–191. Springer, 2009.

GB84. J. A. Goguen and R. M. Burstall. *Introducing institutions*, volume 164 of *Lecture Notes in Computer Science*, chapter chapter 16, pages 221–256. Springer Berlin Heidelberg, 1984.

GB92. Joseph A. Goguen and Rod M. Burstall. Institutions: Abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

GdL12. Esther Guerra and Juan de Lara. An algebraic semantics for QVT-relations check-only transformations. *Fundam. Inform.*, 114(1):73–101, 2012.

GR02. Joseph A. Goguen and Grigore Rosu. Institution morphisms. *Formal Asp. Comput.*, 13(3-5):274–307, 2002.

JKMR13. Phillip James, Alexander Knapp, Till Mossakowski, and Markus Roggenbach. Designing domain specific languages: A craftsman's approach for the railway domain using CASL. In *Recent Trends in Algebraic Development Techniques*, volume 7841 of *Lecture Notes in Computer Science*, pages 178–194. Springer, 2013.

KBA02. I. Kurtev, J. Bezivin, and M. Aksit. Technological spaces: An initial appraisal. In *International Symposium on Distributed Objects and Applications, DOA 2002*, 2002.

Ken02. Stuart Kent. Model-driven engineering. In Michael Butler, Luigia Petre, and Kaisa Sere, editors, *IFM*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer, 2002.

Lan98. Saunders M. Lane. *Categories for the Working Mathematician (Graduate Texts in Mathematics)*. Springer, 2nd edition, September 1998.

LR11. Kevin Lano and Shekoufeh Kolahdouz Rahimi. Model-driven development of model transformations. In Jordi Cabot and Eelco Visser, editors, *ICMT*, volume 6707 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2011.

MHST03. Till Mossakowski, Anne Elisabeth Haxthausen, Donald Sannella, and Andrzej Tarlecki. Casl - the common algebraic specification language: Semantics and proof theory. *Computers and Artificial Intelligence*, 22(3-4):285–321, 2003.

Mos05. Till Mossakowski. Heterogeneous specification and the heterogeneous tool set. Technical report, Universitaet Bremen, 2005. Habilitation thesis.

OMG. Object Management Group. URL: www.omg.org.

OMG03. OMG. Meta Object Facility (MOF) 2.0 Core Specification. Specification Version 2.0, Object Management Group, 2003.

OMG05. OMG. Unified Modeling Language: Superstructure. Specification Version 2.0, Object Management Group, August 2005.

OMG09. OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation. Final Adopted Specification Version 1.1, Object Management Group, 2009.

OMG10. OMG. Object Constraint Language. Formal Specification Version 2.2, Object Management Group, 2010.

OW09. Fernando Orejas and Martin Wirsing. On the specification and verification of model transformations. In Jens Palsberg, editor, *Semantics and Algebraic Specification*, volume 5700 of *Lecture Notes in Computer Science*, pages 140–161. Springer, 2009.

Sch06. Douglas Schmidt. Guest editor's introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.

SMR11. Kurt Stenzel, Nina Moebius, and Wolfgang Reif. Formal verification of QVT transformations for code generation. In Jon Whittle, Tony Clark, and Thomas Kühne, editors, *MoDELS*, volume 6981 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2011.

ST12. Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. EATCS Monographs on theoretical computer science. Springer, 2012.

SZ09. Lijun Shan and Hong Zhu. Semantics of metamodels in UML. In *Proceedings of the 2009 Third IEEE International Symposium on Theoretical Aspects of Software Engineering*, TASE '09, pages 55–62. IEEE Computer Society, 2009.

A Proofs :: Institution for the Conformance Relation

Lemma 1. *Signatures and signature morphisms define a category* Sign*. The points of the category are the signatures and it arrows are the signature morphisms.*

Proof. Let $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i)$ $(i = 1..4)$ *with* $\boldsymbol{T}_i = (T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$ *and* $\boldsymbol{P}_i = (R_i, P_i)$ *be signatures, and let* $\sigma_i : \Sigma_i \rightarrow \Sigma_{i+1}$ $(i{=}1..3)$ *be signature morphisms, then:*

- *Signature morphisms can be composed. We define the composition* $\sigma_2 \circ \sigma_1$ *as the tuple* $\langle \sigma_T, \sigma_R, \sigma_P \rangle$ *such that* $\sigma_T(c) = \sigma_{T_2}(\sigma_{T_1}(c))$, $\sigma_R(c) = \sigma_{R_2}(\sigma_{R_1}(c))$, *and* $\sigma_P(c) = \sigma_{P_2}(\sigma_{P_1}(c))$. *We have to show that* $\sigma_2 \circ \sigma_1$ *is a signature morphism:*
  - *For all* $a \in C_1$ *we have that* $\sigma_T(a) = \sigma_{T_2}(\sigma_{T_1}(c))$ *by definition of* σ_T, *and that* $\sigma_{T_1}(c) \in C_2$ *by definition of* σ_{T_1}. *Moreover,* $\sigma_{T_2}(\sigma_{T_1}(c)) \in C_3$ *by definition of* σ_{T_2}. *In consequence,* $a \in C_1$ *implies* $\sigma_T(a) \in C_3$.
  - *In the same way as before, we conclude that* $a \in T(C_1) \backslash C_1$ *implies* $\sigma_T(a) \in T(C_3) \backslash C_3$.
  - *For all* $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *we have that* $\sigma_{T_1}(a) \leq_{T(C_2)} \sigma_{T_1}(b)$ *by definition of* σ_{T_1}. *Moreover, we have that* $\sigma_{T_1}(a), \sigma_{T_1}(b) \in T(C_2)$ *and thus* $\sigma_{T_2}(\sigma_{T_1}(a)) \leq_{T(C_3)} \sigma_{T_2}(\sigma_{T_1}(b))$ *by definition of* σ_{T_2}. *Finally, we conclude that* $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *implies* $\sigma_T(a) \leq_{T(C_3)} \sigma_T(b)$ *by definition of* σ_T.
  - *For all* $a \in C_1|_{abstract}$ *we have that* $\sigma_{T_1}(a) \in C_2|_{abstract}$ *by definition of* σ_{T_1} *and also that* $\sigma_{T_2}(\sigma_{T_1}(a)) \in C_3|_{abstract}$ *by definition of* σ_{T_2}. *Finally,* $a \in C_1|_{abstract}$ *implies* $\sigma_T(a) \in C_3|_{abstract}$ *by definition of* σ_T.
  - *For all* $p_w \in P_1$ *we have that* $\sigma_{P_1}(p)_{\sigma_1(w)} \in P_2$ *by definition of* σ_{P_1} *and also that* $\sigma_{P_2}(\sigma_{P_1}(p))_{\sigma_2(\sigma_1(w))} \in P_3$ *by definition of* σ_{P_2}. *Finally,* $p_w \in P_1$ *implies* $\sigma_P(p)_{\sigma(w)} \in P_3$ *by definition of* σ_P.
- *Composition of signature morphisms is associative, i.e.* $(\sigma_3 \circ \sigma_2) \circ \sigma_1 = \sigma_3 \circ (\sigma_2 \circ \sigma_1)$*:*
  - *For each* $a \in C_1$ *we have that* $\sigma_{T_2} \circ \sigma_{T_1}(a) = \sigma_{T_2}(\sigma_{T_1}(a))$ *and thus* $\sigma_{T_3} \circ (\sigma_{T_2} \circ \sigma_{T_1})(a) = \sigma_{T_3}(\sigma_{T_2}(\sigma_{T_1}(a)))$ *by the definition of composition. Finally, this last result is equals to* $\sigma_{T_3} \circ \sigma_{T_2}(\sigma_{T_1}(a))$ *which is equals to* $(\sigma_{T_3} \circ \sigma_{T_2}) \circ \sigma_{T_1}(c)$.
  - *The proof is the same in the case of* σ_R *and* σ_P.
- *There exists an identity signature morphism* $id_{\Sigma_1} : \Sigma_1 \rightarrow \Sigma_1$ *defined as a tuple* $\langle id_T, id_R, id_P \rangle$ *such that* $id_T(c) = c$, $id_R(c) = c$, *and* $id_P(c) = c$. *This morphism satisfies the signature morphism conditions:*
  - $a \in C_1$ *implies* $id_T(a) \in C_1$,
  - $a \in T(C_1) \backslash C_1$ *implies* $id_T(a) \in T(C_1) \backslash C_1$,
  - $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *implies* $id_T(a) \leq_{T(C_1)} id_T(b)$,
  - $a \in C_1|_{abstract}$ *implies* $id_T(a) \in C_1|_{abstract}$,
  - $p_w \in P_1$ *implies* $id_P(p)_{id(w)} \in P_1$,

Finally, signatures and signature morphisms define a category. □

Lemma 2. *There is a functor* Sen *giving a set of formulas* ψ *(object in the category* Set*) for each signature* Σ *(object in the category* Sign*), as shown in the definition of a formula, and a function* $\sigma : \text{Sen}(\Sigma_1) \rightarrow \text{Sen}(\Sigma_2)$ *(arrow in the category* Set*) translating formulas for each signature morphism* $\sigma : \Sigma_1 \rightarrow \Sigma_2$ *(arrow in the category* Sign*), as shown in the extension of the signature morphism to formulas.*

Proof. We have to prove that Sen *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By the extension of the signature morphism to formulas, the image of an arrow $\sigma : \Sigma_1 \rightarrow \Sigma_2$ *in the category* Sign *is an arrow* $\sigma : \text{Sen}(\Sigma_1) \rightarrow \text{Sen}(\Sigma_2)$ *in the category* Set*Also, by the definition of formulas, the image of a signature* Σ *in the category* Sign *is an object* $\text{Sen}(\Sigma)$ *in the category* Set*Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that $\text{Sen}(\sigma_2 \circ \sigma_1) = \text{Sen}(\sigma_2) \circ \text{Sen}(\sigma_1)$.
Let Σ_i $(i{=}1..4)$ *be signatures, and let* $\sigma_i{:}\Sigma_i \rightarrow \Sigma_{i+1}$ $(i{=}1, 2)$ *be signature morphisms.* $\text{Sen}(\sigma_2) \circ \text{Sen}(\sigma_1)$ *is the*

canonical application of the signature morphism σ_1 to the elements in ψ_1, composed with the canonical application of the signature morphism σ_2 (by definition of signature morphism extend to formulas). Since signature morphisms can be composed (as defined in Lemma 1), this is the same as the canonical application of the composition of the signature morphism to ψ_1, i.e. $\text{Sen}(\sigma_2 \circ \sigma_1)$.

(c) Let $id_{\Sigma_1} : \Sigma_1 \to \Sigma_1$ be an identity signature morphism (defined in Lemma 1). We can see that identities are preserved since, by definition, for any Σ_1-formula ψ_1, $id_{\Sigma_1}(\psi_1)$ is a Σ_1-formula such that:

- $id(x^c) = x^{id(c)} = x^c$
- $id(\texttt{rel}(p_w, x^c, y^d)) = \texttt{rel}(id_P(p)_{id(w)}, x^{id(c)}, y^{id(d)}) = \texttt{rel}(p_w, x^c, y^d)$
- $id(r \bullet p) = id_R(r) \bullet id_P(p) = r \bullet p$

Finally, the functor Sen *is defined.* □

Lemma 3. *For any signatures, the Σ-interpretations and Σ-homomorphisms define a category* $\text{Mod}(\Sigma)$*. The points of the category are the Σ-interpretations, its arrows are the Σ-homomorphisms.*

Proof. Let $\Sigma = (\boldsymbol{T}, \boldsymbol{P})$ with $\boldsymbol{T} = (T(C), \leq_{T(C)}, C|_{abstract})$ and $\boldsymbol{P} = (R, P)$ be a signature, let $\mathcal{I}_i = (\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}_i), \boldsymbol{A}_i)$ $(i{=}1..4)$ be Σ-interpretations, and let $h_i : \mathcal{I}_i \to \mathcal{I}_{i+1}$ $(i{=}1..3)$ be Σ-homomorphisms, then:

- *Σ-homomorphisms can be composed. We define the composition $h_2 \circ h_1$ as a a family of maps $(h_c)_{c \in T(C)}$ with $h_c : V_{c_1} \to V_{c_3}$ such that: $h_c(v) = h_{c_2}(h_{c_1}(v))$, $v \in O_{c_1}$. We have to prove that $h_2 \circ h_1$ is a Σ-homomorphism, i.e.*
  - *$h_c(v) \in O_{c_3}$ forall $v \in O_{c_1}$. By definition of homomorphism, we have that for each $v \in O_{c_1}$, $h_{c_1}(v) \in O_{c_2}$ and that for each $w \in O_{c_2}$ $h_{c_2}(w) \in O_{c_3}$, thus $h_{c_2}(h_{c_1}(v)) = h_c(v) \in O_{c_3}$.*
  - *$h_c(v) \in V_{c_3} \backslash O_{c_3}$ forall $v \in V_{c_1} \backslash O_{c_1}$. Proceeding in the same way as before, we have that for each $v \in V_{c_1} \backslash O_{c_1}$, $h_{c_2}(h_{c_1}(v)) = h_c(v) \in V_{c_3} \backslash O_{c_3}$.*
  - *$(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_c(v_1), h_c(v_2)) \in p^{\mathcal{I}_3}$ for any $v_i \in V_{c_1}$ $(i = 1, 2)$ and $p(r_1 : c_1, r_2 : c_2) \in P$. By definition of homomorphism, we have that for any $v_i \in V_{c_1}$ $(i = 1, 2)$ and $p(r_1 : c_1, r_2 : c_2) \in P$, $(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_{c_1}(v_1), h_{c_2}(v_2)) \in p^{\mathcal{I}_2}$, and also for any $w_i \in V_{c_2}$ $(i = 1, 2)$ and $p(r_1 : c_1, r_2 : c_2) \in P$, $(w_1, w_2) \in p_2^{\mathcal{I}}$ iff $(h_{c_2}(w_1), h_{c_2}(w_2)) \in p^{\mathcal{I}_2}$. Thus, for any $v_i \in V_{c_1}$ $(i = 1, 2)$ and $p(r_1 : c_1, r_2 : c_2) \in P$, $(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_{c_2}(h_{c_1}(v_1)), h_{c_2}(h_{c_1}(v_2))) = (h_c(v_1), h_c(v_2)) \in p^{\mathcal{I}_3}$.*
- *Composition of Σ-homomorphisms is associative, i.e., $(h_3 \circ h_2) \circ h_1 = h_3 \circ (h_2 \circ h_1)$. By definition of composition of homomorphisms, for each $v \in O_{c_1}$ we have that $(h_3 \circ h_2) \circ h_1(v) = (h_3 \circ h_2)(h_{c_1}(v)) = h_{c_3}(h_{c_2}(h_{c_1}(v))) = h_3 \circ h_{c_2}(h_{c_1}(v)) = h_3 \circ (h_2 \circ h_1)$.*

- *There exist an identity Σ-homomorphism $id_{\mathcal{I}_1} : \mathcal{I}_1 \to \mathcal{I}_1$ consisting of a family of maps $(id_c)_{c \in T(C)}$ with $id_c : V_{c_1} \to V_{c_1}$ such that: $id_c(v) = v$, $v \in O_{c_1}$. It trivially holds that $id_{\mathcal{I}_1}$ is a Σ-homomorphism since:*
  - *$id_c(v) = v \in O_{c_1}$ forall $v \in O_{c_1}$.*
  - *$id_c(v) = v \in V_{c_1} \backslash O_{c_1}$ forall $v \in V_{c_1} \backslash O_{c_1}$.*
  - *$(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(id_c(v_1), id_c(v_2)) = (v_1, v_2) \in p^{\mathcal{I}_1}$ for any $v_i \in V_{c_1}$ $(i = 1, 2)$ and $p(r_1 : c_1, r_2 : c_2) \in P$.*

Finally, Σ-interpretations and Σ-homomorphisms define a category. □

Lemma 4. *The reduct of Σ-interpretations and Σ-homomorphisms is a functor* $\text{Mod}(\sigma)$ *from Σ_2-interpretations to Σ_1-interpretations (and Σ_2-homomorphisms to Σ_1-homomorphisms) for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$.*

Proof. By definition, domain and codomain of the reduct of an Σ-homomorphism are the reduct of domain and codomain, respectively, of the Σ-homomorphism. We have now to prove that: (a) the reduct of a composition of two Σ-homomorphisms is the composition of the reducts of those Σ-homomorphisms, and (b) that the reduct of an identity Σ-homomorphisms is likewise an identity.

Let $\Sigma = (\boldsymbol{T}, \boldsymbol{P})$ with $\boldsymbol{T} = (T(C), \leq_{T(C)}, C|_{abstract})$ and $\boldsymbol{P} = (R, P)$ be a signature, let $\mathcal{I}_i = (\boldsymbol{V_C^T}(\boldsymbol{O}_i), \boldsymbol{A}_i)$ $(i{=}1..3)$ be Σ-interpretations, and let $h_i : \mathcal{I}_i \to \mathcal{I}_{i+1}$ $(i{=}1, 2)$ be Σ-homomorphisms.

(a) $(h_2 \circ h_1)|_\sigma = h_2|_\sigma \circ h_1|_\sigma$. By definition of reduct of homomorphisms, we have that for any $c \in T(C)$, for any $v \in V_c$, $(h_2 \circ h_1)|_\sigma$ is defined by $(h_2 \circ h_1)_{\sigma(c)}(v)$. By definition of composition of homomorphisms, this is equals to $(h_2)_{\sigma(c)}((h_1)_{\sigma(c)}(v))$. Thus, by definition of composition of homomorphisms, this is equals to $((h_2)_{\sigma(c)} \circ (h_1)_{\sigma(c)})(v)$ which is the definition of $h_2|_\sigma \circ h_1|_\sigma$.

(b) Let $id_{\mathcal{I}_2}$ be an identity Σ_2-homomorphism, then $id_{\mathcal{I}_2}|_\sigma$ is an identity Σ_1-homomorphism, since by definition of reduct of a homomorphism $id_{\mathcal{I}_2}|_\sigma$ is the Σ_1-homomorphism h_1 defined by $h_{1_c}(v) = id_{\mathcal{I}_{2_{\sigma(c)}}}(v) = v$ for any $c \in T(C)$, for any $v \in V_c$.

Finally, the reduct of Σ-interpretations and Σ-homomorphisms is a functor. □

Lemma 5. *There is a functor* Mod *giving a category* Mod*(Σ) of Σ-interpretations (object in the category* **Cat***) for each signature Σ (object in the category* Sign*), as shown in Lemma 3, and a functor* Mod*(σ) (arrow in the category* **Cat***) from Σ_2-interpretations to Σ_1-interpretations (and Σ_2-homomorphisms to Σ_1-homomorphisms) for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ (arrow in the category* Sign*), as shown in Lemma 4.*

Proof. We have to prove that Mod *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By Lemma 4, the image of an arrow $\sigma : \Sigma_2 \to \Sigma_1$ in the category Sign^{op} *is an arrow* Mod(σ) : Mod$(\Sigma_2) \to$ Mod(Σ_1) *in the category* **Cat***. Also, by Lemma 3, the image of a signature Σ in the category* Sign *is an object* Mod*(Σ) in the category* **Cat***. Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that Mod$(\sigma_2 \circ \sigma_1)$ = Mod$(\sigma_2) \circ$ Mod(σ_1) *for both, interpretations and homomorphisms. Let Σ_i $(i{=}1..3)$ be signatures, let $\sigma_i : \Sigma_i \to \Sigma_{i+1}$ $(i{=}1, 2)$ be signature morphisms, let $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$ be a Σ_3-interpretation, and let h be a Σ_3-homomorphism. Then, we have to prove:*

- $\mathcal{I}|_{\sigma_2 \circ \sigma_1} = (\mathcal{I}|_{\sigma_2})|_{\sigma_1}$.
  By definition of reduct, $\mathcal{I}|_{\sigma_2}$ is the Σ_2-interpretation $(\boldsymbol{V_C^T}(\boldsymbol{O}|_{\sigma_2}), \boldsymbol{A}|_{\sigma_2})$ such that:
  - $\boldsymbol{V_C^T}(\boldsymbol{O}|_{\sigma_2}) = (V_{\sigma_2(c)})_{c \in T(C_2)}$
  - $\boldsymbol{A}|_{\sigma_2} = \{\sigma_{2p}(p)^{\mathcal{I}} \mid p \in P_2\}$

  Then $(\mathcal{I}|_{\sigma_2})|_{\sigma_1}$ is the Σ_1-interpretation $(\boldsymbol{V_C^T}((\boldsymbol{O}|_{\sigma_2})|_{\sigma_1}), (\boldsymbol{A}|_{\sigma_2})|_{\sigma_1})$ such that:
  - $\boldsymbol{V_C^T}((\boldsymbol{O}|_{\sigma_2})|_{\sigma_1}) = (V_{\sigma_2(\sigma_1(c))})_{c \in T(C_1)}$
  - $(\boldsymbol{A}|_{\sigma_2})|_{\sigma_1} = \{\sigma_{2p}(\sigma_{1p}(p))^{\mathcal{I}} \mid p \in P_1\}$

  and this is equal to $\mathcal{I}|_{\sigma_2 \circ \sigma_1}$.
- $h|_{\sigma_2 \circ \sigma_1} = (h|_{\sigma_2})|_{\sigma_1}$.
  By definition of reduct, $h|_{\sigma_2}$ is defined by $h|_{\sigma_2 c}(v) = h_{\sigma_2(c)}(v)$ for any $c \in T(C_2)$, for any $v \in V_c$, and thus $(h|_{\sigma_2})|_{\sigma_1}$ is defined by $(h|_{\sigma_2})|_{\sigma_1 c}(v) = h_{\sigma_2(\sigma_1(c))}(v) = h_{\sigma_2 \circ \sigma_1(c)}(v)$ for any $c \in T(C_1)$, for any $v \in V_c$, which is equals to $h|_{\sigma_2 \circ \sigma_1}$.

(c) Let $id_\sigma : \Sigma \to \Sigma$ be an identity signature morphism (defined in Lemma 1). We have to prove that Mod(id_σ) *is an identity functor, i.e., it is composed by the identity reduct of Σ-interpretations and the identity reduct of Σ-homomorphisms.*

- *By definition of reduct, for any Σ-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$, $\mathcal{I}|_{id_\sigma}$ is the Σ-interpretation $(\boldsymbol{V_C^T}(\boldsymbol{O}|_{id_\sigma}), \boldsymbol{A}|_{id_\sigma})$ such that:*

- $\boldsymbol{V_C^T}(\boldsymbol{O}|_{id_\sigma}) = (V_{id_\sigma(c)})_{c\in T(C)}$
- $\boldsymbol{A}|_{id_\sigma} = \{id_{\sigma p}(p)^{\mathcal{I}} \mid p \in P\}$

Finally, by the definition of id_σ, $\mathcal{I}|_{id_\sigma} = \mathcal{I}$, thus $\_|_{id_\sigma}$ is the identity reduct of Σ-interpretations.

– *By definition of reduct, given a Σ-interpretation $\mathcal{I}_1 = (\boldsymbol{V_C^T}(\boldsymbol{O}_1), \boldsymbol{A}_1)$, for any Σ-homomorphism $h : \mathcal{I}_1 \to \mathcal{I}_2$, the reduct $h|_{id_\sigma}$ is defined by $h|_{id_\sigma\, c}(v) = h_{id_\sigma(c)}(v) = h_c(v)$ for any $c \in T(C)$, for any $v \in V_c$. Now, since $\mathcal{I}|_{id_\sigma} = \mathcal{I}$, we have that $\_|_{id_\sigma}$ is the identity reduct of Σ-homomorphisms.*

Finally, the functor Mod *is defined.* □

Theorem 1 (Satisfaction Condition). *Given signatures $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i)$ $(i = 1, 2)$ with $\boldsymbol{T}_i = (T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$ and $\boldsymbol{P}_i = (R_i, P_i)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A})$, a Σ_2-valuation $\beta = (\beta^c)_{c\in T(C_2)}$, and a Σ_1-formula $\psi = (Obj, Rel, \Phi)$ with variables $X = (X^c)_{c\in T(C_1)}$, the following satisfaction condition holds.*

$$\mathcal{I}|_\sigma, \beta|_\sigma \models_{\Sigma_1} \psi \text{ iff } \mathcal{I}, \beta \models_{\Sigma_2} \sigma(\psi)$$

Proof. By definition of the satisfaction relation, we need to prove that:

(1) $\forall\ \mathtt{rel}(p, x^c, y^d) \in Rel,\ \exists\ p^{\mathcal{I}|_\sigma}((\beta|_\sigma)^c(x^c), (\beta|_\sigma)^d(y^d)) \in \boldsymbol{A}|_\sigma$
$\Leftrightarrow\ \forall\ \mathtt{rel}(\sigma_P(p), x^{\sigma(c)}, y^{\sigma(d)}) \in \sigma(Rel),\ \exists\ \sigma_P(p)^{\mathcal{I}}(\beta^{\sigma(c)}(x^{\sigma(c)}), \beta^{\sigma(d)}(y^{\sigma(d)})) \in \boldsymbol{A}$

In the one side we have that $\mathtt{rel}(p, x^c, y^d) \in Rel \Leftrightarrow \mathtt{rel}(\sigma_P(p), x^{\sigma(c)}, y^{\sigma(d)}) \in \sigma(Rel)$ by definition of σ. In the other side we have that $p^{\mathcal{I}|_\sigma}((\beta|_\sigma)^c(x^c), (\beta|_\sigma)^d(y^d)) \in \boldsymbol{A}|_\sigma \Leftrightarrow \sigma_P(p)^{\mathcal{I}}(\beta^{\sigma(c)}(x^{\sigma(c)}), \beta^{\sigma(d)}(y^{\sigma(d)})) \in \boldsymbol{A}$, since $p^{\mathcal{I}|_\sigma} = \sigma_P(p)^{\mathcal{I}}\ \forall p \in P_1$ by definition of $|_\sigma$, and $(\beta\,|_\sigma)^c(x^c) = \beta^{\sigma(c)}(x^{\sigma(c)})$ and $(\beta\,|_\sigma)^d(y^d) = \beta^{\sigma(d)}(y^{\sigma(d)})$ by definition of $\beta|_\sigma$. Finally, using both results we can conclude that (1) holds.

(2) $\forall\ \varphi \in \Phi.\ \mathcal{I}|_\sigma|_{(\varphi,\beta|_\sigma)} \models_{\Sigma_1} \varphi \Leftrightarrow \forall\ \sigma(\varphi) \in \sigma(\Phi).\ \mathcal{I}|_{(\varphi,\beta)} \models_{\Sigma_2} \sigma(\varphi)$

We know that $p^{\mathcal{I}|_\sigma} = \sigma_P(p)^{\mathcal{I}}\ \forall p(r_1 : c, r_2 : d) \in P_1$ by definition of $|_\sigma$. Also, if $p^{\mathcal{I}|_\sigma}$ is the corresponding interpretation of a formula $\mathtt{rel}(p, x^c, y^d)$, we have that $p^{\mathcal{I}|_\sigma|_{(\varphi,\beta|_\sigma)}} = \sigma_P(p)^{\mathcal{I}|_{(\sigma(\varphi),\beta)}}$. With this result we can deduce that $(r \bullet p)^{\mathcal{I}|_\sigma|_{(\varphi,\beta|_\sigma)}} = (\sigma_R(r), \sigma_P(p))^{\mathcal{I}|_{(\sigma(\varphi),\beta)}}$. Moreover, for all φ of the form $\#(r \bullet p) = n$, we have that $\sigma(\varphi)$ is $\#(\sigma_R(r) \bullet \sigma_P(p)) = n$, by definition of σ. Finally, using both results we have that φ is $\#(r \bullet p) = n$ and $|S| = n\ \forall\ S \in (r \bullet p)^{\mathcal{I}|_\sigma|_{(\varphi,\beta|_\sigma)}} \Leftrightarrow \sigma(\varphi)$ is $\#(\sigma_R(r) \bullet \sigma_P(p)) = n$ and $|S| = n\ \forall\ S \in (\sigma_R(r), \sigma_P(p))^{\mathcal{I}|_{(\sigma(\varphi),\beta)}}$. Thus, $\mathcal{I}|_\sigma|_{(\varphi,\beta|_\sigma)} \models_{\Sigma_1} \varphi \Leftrightarrow \mathcal{I}|_{(\varphi,\beta)} \models_{\Sigma_2} \sigma(\varphi)$. We can proceed exactly in the same way for proving the other two cases of φ.

Finally, with (1) and (2) we conclude that the satisfaction condition holds. □

B Proofs :: Institution for Model Transformations

Lemma 6. *Signatures and signature morphisms define a category* Sign*. The points of the category are the signatures and it arrows are the signature morphisms.*

Proof. A signature morphisms defined as a triple of morphisms of the corresponding institutions $\langle \sigma_1^{\mathrm{C}}, \sigma_2^{\mathrm{C}}, \sigma^{\mathrm{FOL}} \rangle$*. In those institutions, morphisms are composable, the composition is associative, and there exists an identity signature morphism. We define the composition* $\sigma_2 \circ \sigma_1$ *as the componentwise composition of morphisms, as well as the identity signature morphism as the triple with the identity signature morphisms of the corresponding institutions. Using these facts, it is straightforward to conclude that: signature morphisms can be composed, composition of signature morphisms is associative, and there exists an identity signature morphism. Finally, signatures and signature morphisms define a category.* □

Lemma 7. *There is a functor* Sen *giving a set of formulas* ψ *(object in the category* Set*) for each signature* Σ *(object in the category* Sign*), as shown in the definition of a formula, and a function* $\sigma : \mathrm{Sen}(\Sigma_1) \to \mathrm{Sen}(\Sigma_2)$ *(arrow in the category* Set*) translating formulas for each signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$ *(arrow in the category* Sign*), as shown in the extension of the signature morphism to formulas.*

Proof. We have to prove that Sen *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By the extension of the signature morphism to formulas, the image of an arrow $\sigma : \Sigma_1 \to \Sigma_2$ *in the category* Sign *is an arrow* $\sigma : \mathrm{Sen}(\Sigma_1) \to \mathrm{Sen}(\Sigma_2)$ *in the category* Set*Also, by the definition of formulas, the image of a signature* Σ *in the category* Sign *is an object* $\mathrm{Sen}(\Sigma)$ *in the category* Set*Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that $\mathrm{Sen}(\sigma_2 \circ \sigma_1) = \mathrm{Sen}(\sigma_2) \circ \mathrm{Sen}(\sigma_1)$.
Let Σ_i $(i{=}1..4)$ *be signatures, and let* $\sigma_i{:}\Sigma_i \to \Sigma_{i+1}$ $(i{=}1,2)$ *be signature morphisms.* $\mathrm{Sen}(\sigma_2) \circ \mathrm{Sen}(\sigma_1)$ *is the canonical application of the signature morphism* σ_1 *to the elements in* ψ_1*, composed with the canonical application of the signature morphism* σ_2 *(by definition of signature morphism extend to formulas). Since signature morphisms can be composed (as defined in Lemma 6), this is the same as the canonical application of the composition of the signature morphism to* ψ_1*, i.e.* $\mathrm{Sen}(\sigma_2 \circ \sigma_1)$.

(c) Let $id_{\Sigma_1} : \Sigma_1 \to \Sigma_1$ *be an identity signature morphism (defined in Lemma 6). We can see that identities are preserved since is the componentwise application of the identity signature morphisms to every element in the formula, which already preserve the identities.*

Finally, the functor Sen *is defined.* □

Lemma 8. *For any signatures, the* Σ*-models and* Σ*-homomorphisms define a category* Mod*(*Σ*). The points of the category are the* Σ*-interpretations, its arrows are the* Σ*-homomorphisms.*

Proof. A Σ*-model is defined as a triple of* Σ*-models of the corresponding institutions* $\langle \mathcal{M}_1^{\mathrm{C}}, \mathcal{M}_2^{\mathrm{C}}, \mathcal{M}^{\mathrm{FOL}} \rangle$*, as well as homomorphisms are defined componentwise* $\langle h_1^{\mathrm{C}}, h_2^{\mathrm{C}}, h^{\mathrm{FOL}} \rangle$*. In the corresponding institutions, homomorphisms are composable, the composition is associative, and there exists an identity homomorphism. We define the composition of homomorphisms as the componentwise composition of homomorphisms, as well as the identity homomorphism as the triple with the identity homomorphisms of the corresponding institutions. Using these facts, it is straightforward to conclude that: homomorphisms can be composed, composition of homomorphisms is associative, and there exists an identity homomorphism. Finally,* Σ*-interpretations and* Σ*-homomorphisms define a category.* □

Lemma 9. *The reduct of* Σ*-models and* Σ*-homomorphisms is a functor* Mod*(*σ*) from* Σ_2*-models to* Σ_1*-models (and* Σ_2*-homomorphisms to* Σ_1*-homomorphisms) for each signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$*.*

Proof. A Σ-model is defined as a triple of Σ-models of the corresponding institutions $\langle \mathcal{M}_1^{\mathrm{C}}, \mathcal{M}_2^{\mathrm{C}}, \mathcal{M}^{\mathrm{FOL}} \rangle$, as well as reducts are defined componentwise $\langle \mathcal{M}_1^{\mathrm{C}}|_\sigma, \mathcal{M}_2^{\mathrm{C}}|_\sigma, \mathcal{M}^{\mathrm{FOL}}|_\sigma \rangle$. In the corresponding institutions, the reduct of models and homomorphisms is a functor. From this we can conclude straightforward that domain and codomain of the reduct of an Σ-homomorphism are the reduct of domain and codomain, respectively, the reduct of a composition of two Σ-homomorphisms is the composition of the reducts of those Σ-homomorphisms, and the reduct of an identity Σ-homomorphisms is likewise an identity. Finally, the reduct of Σ-interpretations and Σ-homomorphisms is a functor. □

Lemma 10. *There is a functor* Mod *giving a category* Mod*(Σ) of Σ-models (object in the category* **Cat***) for each signature Σ (object in the category* Sign*), as shown in Lemma 8, and a functor* Mod*(σ) (arrow in the category* **Cat***) from Σ_2-models to Σ_1-models (and Σ_2-homomorphisms to Σ_1-homomorphisms) for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ (arrow in the category* Sign*), as shown in Lemma 9.*

Proof. We have to prove that Mod *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By Lemma 9, the image of an arrow $\sigma : \Sigma_2 \to \Sigma_1$ in the category Sign^{op} *is an arrow* $\mathrm{Mod}(\sigma) : \mathrm{Mod}(\Sigma_2) \to \mathrm{Mod}(\Sigma_1)$ *in the category* **Cat**. *Also, by Lemma 8, the image of a signature Σ in the category* Sign *is an object* Mod*(Σ) in the category* **Cat**. *Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that $\mathrm{Mod}(\sigma_2 \circ \sigma_1) = \mathrm{Mod}(\sigma_2) \circ \mathrm{Mod}(\sigma_1)$ *for both, models and homomorphisms. A reduct or homomorphism is defined as a triple of reducts or homomorphisms, respectively, of the corresponding institutions which are applied to each component in the model. Since the property holds in isolation for each component, we can directly conclude that this also holds for the triple.*

(c) Let $id_\sigma : \Sigma \to \Sigma$ be an identity signature morphism (defined in Lemma 1). We have to prove that $\mathrm{Mod}(id_\sigma)$ *is an identity functor, i.e., it is composed by the identity reduct of Σ-models and the identity reduct of Σ-homomorphisms. Since reducts and homomorphisms are defined componentwise and this property holds in isolation for each component, we can directly conclude that this also holds for the triple.*

Finally, the functor Mod *is defined.* □

Theorem 2 (Satisfaction Condition). *Given signatures $\Sigma_i = \langle \Sigma_1^{\mathrm{C}_i}, \Sigma_2^{\mathrm{C}_i}, \Sigma^{\mathrm{FOL}_i} \rangle (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M} = \langle \mathcal{M}_1^{\mathrm{C}_2}, \mathcal{M}_2^{\mathrm{C}_2}, \mathcal{M}^{\mathrm{FOL}_2} \rangle$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, and a Σ_1-formula $\varphi = \langle \varphi_1^{\mathrm{C}_1}, \varphi_2^{\mathrm{C}_1}, \varphi^{\mathrm{rules}} \rangle$ with variables in $X_2|_\sigma$, the following satisfaction condition holds.*

$$\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi \text{ iff } \mathcal{M} \models_{\Sigma_2} \sigma(\varphi)$$

Proof. We will first prove some preliminary results:

1. *Given $FOL^=$ signatures Σ_i $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-first-order structure $\mathcal{M}$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, and a Σ_1-formula φ with variables in $X_2|_\sigma$, the following condition holds by definition of the $FOL^=$ institution: $\mathcal{M}|_\sigma, \mu|_\sigma \models_{\Sigma_1} \varphi$ iff $\mathcal{M}, \mu \models_{\Sigma_2} \sigma(\varphi)$*

2. *Given signatures $\Sigma_i (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M}$ with a Σ_2-first-order structure $\mathcal{M}^{\mathrm{FOL}}$, and a Σ_1-formula φ, the following condition holds: $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi = \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}|_\sigma$*

   Proof. In one side, $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi$ is a Σ_2-structure which is taken for interpreting element in Σ_1 and then restricted for those elements in φ. In the other side, $\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}|_\sigma$ is the same structure with an interpretation for those

elements in $\sigma(\varphi)$ (which are the same as φ up to the signature morphism σ) and then taken to interpret elements in Σ_1 (as in the first case). In conclusion, both Σ_1-structures have the minimal semantic element needed for interpreting syntactic elements in φ, thus there are equal.

3. *Given signatures $\Sigma_i (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M}$ with a Σ_2-first-order structure $\mathcal{M}^{\mathrm{FOL}}$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, a Σ_1-formula φ with variables in $X_2|_\sigma$, and a pattern* $\mathrm{Pattern} = \langle E, A, Pr \rangle$ *which is part of φ, the following condition holds:*
$\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models \sigma(\mathrm{Pattern})$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models \mathrm{Pattern}$

Proof.

$$
\begin{array}{ll}
\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models \sigma(\mathrm{Pattern}) & \\
\quad \textit{iff } \sigma(p)_D(\mu(\sigma(x)), \mu(\sigma(y))) \in \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)} \,.\, \forall\, rel(\sigma(p), \sigma(x), \sigma(y)) \in \sigma(A) & \\
\quad\quad \textit{and } \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models_{\mathrm{FOL}} \sigma(Pr) & \textit{by def. of pattern satisf.} \\
\quad \textit{iff } p_D(\mu|_\sigma\,(x), \mu|_\sigma\,(y)) \in \mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi \,.\, \forall\, rel(p, x, y) \in A & \textit{by def. of } \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)} \textit{ and } \sigma \\
\quad\quad \textit{and } \mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models_{\mathrm{FOL}} Pr & \textit{by result 1 and result 2}
\end{array}
$$

Finally, by definition of satisfaction of a pattern, we conclude that $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models \mathrm{Pattern}$ *also holds.*

4. *Given signatures $\Sigma_i (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M}$ with a Σ_2-first-order structure $\mathcal{M}^{\mathrm{FOL}}$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, a Σ_1-formula φ with variables in $X_2|_\sigma$, and a* `when` *clause* $\mathrm{when} = \langle \mathrm{when}_c, \mathrm{when}_r \rangle$ *which is part of φ, the following condition holds:*
$\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models \sigma(\mathrm{when})$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models \mathrm{when}$

Proof.

$$
\begin{array}{ll}
\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models \sigma(\mathrm{when}) & \\
\quad \textit{iff } \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models_{\mathrm{FOL}} \sigma(\mathrm{when}_c) & \\
\quad\quad \textit{and } \mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu[\sigma(v)] \models \sigma(r).\, \forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r) & \textit{by def. of satisfaction of a when clause}
\end{array}
$$

We also know that $\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models_{\mathrm{FOL}} \sigma(\mathrm{when}_c)$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models_{\mathrm{FOL}} \mathrm{when}_c$ *by result 1 and result 2. Thus, we have to prove that:*

$$\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu[\sigma(v)] \models \sigma(r).\, \forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r) \textit{ iff } \mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma\, [v] \models r.\, \forall(r, v) \in \mathrm{when}_r$$

This can be proved by induction on the length of the chain of dependencies of `when` *and* `where` *clauses which is assumed to be finite as we discussed before. This means that the base case is* $\mathrm{when}_r = \emptyset$ *in which the condition trivially holds, since also* $\sigma(\mathrm{when}_r) = \emptyset$. *The inductive hypothesis is such that* $\forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r)$ *we have that* $\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu[\sigma(v)] \models \sigma(r)$. *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma\, [v] \models r$ *(and the same* $\forall(r, v) \in \mathrm{when}_r$. *Thus, the inductive thesis trivially holds from these hypothesis.*
Finally, by definition of satisfaction of a `when` *clause, we conclude that* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models \mathrm{when}$ *also holds.*

5. *Given signatures $\Sigma_i (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M}$ with a Σ_2-first-order structure $\mathcal{M}^{\mathrm{FOL}}$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, a Σ_1-formula φ with variables in $X_2|_\sigma$, and a* `where` *clause* $\mathrm{where} = \langle \mathrm{where}_c, \mathrm{where}_r \rangle$ *which is part of φ, the following condition holds:*
$\mathcal{M}^{\mathrm{FOL}}|_{\sigma(\varphi)}, \mu \models \sigma(\mathrm{where})$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma|_\varphi, \mu|_\sigma \models \mathrm{where}$

Proof. The proof is similar to the case of a `when` *clause.*

Now, we can prove the satisfaction condition as follows:

$$
\begin{array}{ll}
\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi & \\
\quad \textit{iff } \mathcal{M}_i^{\mathrm{C}_2}|_\sigma \models^{\mathrm{C}}_{\Sigma_i^{\mathrm{C}_1}} \varphi_i^{\mathrm{C}_1}\ (i = 1, 2) & \\
\quad\quad \textit{and } \mathcal{M}|_\sigma|_\varphi \models_{\Sigma_1} \varphi^{\mathrm{rules}} & \textit{by def. of satisfaction relation} \\
\quad \textit{iff } \mathcal{M}_i^{\mathrm{C}_2} \models^{\mathrm{C}}_{\Sigma_i^{\mathrm{C}_2}} \sigma(\varphi_i^{\mathrm{C}_1})\ (i = 1, 2) & \textit{by def. of } \mathcal{I}^{\mathrm{C}} \\
\quad\quad \textit{and } \mathcal{M}|_{\sigma(\varphi)} \models_{\Sigma_2} \sigma(\varphi^{\mathrm{rules}}) & \textit{as proved next} \\
\quad \textit{iff } \mathcal{M} \models_{\Sigma_2} \sigma(\varphi) & \textit{by def. of satisfaction relation}
\end{array}
$$

We need to prove that $\mathcal{M}\,|_\sigma|_\varphi \models_{\Sigma_1} \varphi^{\mathrm{rules}}$ *iff* $\mathcal{M}\,|_{\sigma(\varphi)} \models_{\Sigma_2} \sigma(\varphi^{\mathrm{rules}})$, *which assuming that* φ^{rules} *is of the form* $\langle \mathrm{Rules}, \mathrm{top} \rangle$, *it is the same to prove that*

$$\forall\ \text{Rule}_i \in \text{top}.\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi, \emptyset \models \text{Rule}_i \textit{ iff } \forall\ \sigma(\text{Rule}_i) \in \sigma(\text{top}).\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)}, \emptyset \models \sigma(\text{Rule}_i)$$

For any rule $\text{Rule} = \langle \text{VarSet}, \text{Pattern}_i\ (i = 1, 2), \text{when}, \text{where} \rangle$*, we have two cases:*

1. *If* $\text{WhenVarSet} = \emptyset$*,* $\mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi, \emptyset \models \text{Rule}$ *if*

$$\begin{array}{l}
\forall\ \mu^1|_\sigma\ [x_1, ..., x_n] \in\ |\text{VarSet}\backslash\text{2_VarSet}|, \\
\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma\ [x_1, ..., x_n] \models \text{Pattern}_1 \rightarrow \\
\quad\quad \exists\ \mu^2|_\sigma\ [y_1, ..., y_m] \in |\text{2_VarSet}|, \\
\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma \cup \mu^2|_\sigma \models \text{Pattern}_2 \wedge \\
\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma \cup \mu^2|_\sigma \models \text{where}))
\end{array}$$

2. *If* $\text{WhenVarSet} \neq \emptyset$*,* $\mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi, \emptyset \models \text{Rule}$ *if*

$$\begin{array}{l}
\forall\ \mu^w|_\sigma\ [z_1, ..., z_o] \in\ |\text{WhenVarSet}|, \\
\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi, \mu^w|_\sigma\ [z_1, ..., z_o] \models \text{when} \rightarrow \\
\quad\quad \forall\ \mu^1|_\sigma\ [x_1, ..., x_n] \in\ |\text{VarSet}\backslash(\text{WhenVarSet} \cup \text{2_VarSet})|, \\
\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma \cup \mu^w|_\sigma \models \text{Pattern}_1 \rightarrow \\
\quad\quad\quad\quad \exists\ \mu^2|_\sigma\ [y_1, ..., y_m] \in |\text{2_VarSet}|, \\
\quad\quad\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma \cup \mu^2|_\sigma \cup \mu^w|_\sigma \models \text{Pattern}_2 \wedge \\
\quad\quad\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_\sigma|_\varphi,\ \mu^1|_\sigma \cup \mu^2|_\sigma \cup \mu^w|_\sigma \models \text{where})))
\end{array}$$

In both cases we can directly use the preliminar results, plus the definition of $\mu|_\sigma$*, to conclude that the following cases also hold:*

1. *If* $\sigma(\text{WhenVarSet}) = \emptyset$

$$\begin{array}{l}
\forall\ \mu^1[x_1, ..., x_n] \in\ |\text{VarSet}\backslash\text{2_VarSet}|, \\
\quad (\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1[x_1, ..., x_n] \models \sigma(\text{Pattern}_1) \rightarrow \\
\quad\quad \exists\ \mu^2[y_1, ..., y_m] \in |\text{2_VarSet}|, \\
\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1 \cup \mu^2 \models \sigma(\text{Pattern}_2) \wedge \\
\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1 \cup \mu^2 \models \sigma(\text{where})))
\end{array}$$

2. *If* $\sigma(\text{WhenVarSet}) \neq \emptyset$

$$\begin{array}{l}
\forall\ \mu^w[z_1, ..., z_o] \in\ |\text{WhenVarSet}|, \\
\quad (\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)}, \mu^w[z_1, ..., z_o] \models \sigma(\text{when}) \rightarrow \\
\quad\quad \forall\ \mu^1[x_1, ..., x_n] \in\ |\text{VarSet}\backslash(\text{WhenVarSet} \cup \text{2_VarSet})|, \\
\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1 \cup \mu^w \models \sigma(\text{Pattern}_1) \rightarrow \\
\quad\quad\quad\quad \exists\ \mu^2[y_1, ..., y_m] \in |\text{2_VarSet}|, \\
\quad\quad\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1 \cup \mu^2 \cup \mu^w \models \sigma(\text{Pattern}_2) \wedge \\
\quad\quad\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)},\ \mu^1 \cup \mu^2 \cup \mu^w \models \sigma(\text{where}))))
\end{array}$$

Finally, we conclude that $\forall\ \sigma(\text{Rule}_i) \in \sigma(\text{top}).\ \mathcal{M}^{\text{FOL}_2}|_{\sigma(\varphi)}, \emptyset \models \sigma(\text{Rule}_i)$*. Note that we can also read this demonstration downside up, thus the satisfaction condition holds.* □

C Proofs :: Alternative Institution for MOF and QVT

Lemma 11. *Signatures and signature morphisms define a category* Sign*. The points of the category are the signatures and it arrows are the signature morphisms.*

Proof. Let $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i, \boldsymbol{M}_i)$ $(i = 1..4)$ *with* $\boldsymbol{T}_i = (T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$, $\boldsymbol{P}_i = (R_i, P_i)$, *and* $\boldsymbol{M}_i = (I_i, L_i)$ *be signatures, and let* $\sigma_i : \Sigma_i \to \Sigma_{i+1}$ $(i{=}1..3)$ *be signature morphisms, then:*

- *Signature morphisms can be composed. We define the composition* $\sigma_2 \circ \sigma_1$ *as the tuple* $\langle \sigma_T, \sigma_R, \sigma_P, \sigma_I \rangle$ *such that* $\sigma_T(c) = \sigma_{T_2}(\sigma_{T_1}(c))$, $\sigma_R(c) = \sigma_{R_2}(\sigma_{R_1}(c))$, $\sigma_P(c) = \sigma_{P_2}(\sigma_{P_1}(c))$, *and* $\sigma_I(c) = \sigma_{I_2}(\sigma_{I_1}(c))$. *We have to show that* $\sigma_2 \circ \sigma_1$ *is a signature morphism:*
  - *For all* $a \in C_1$ *we have that* $\sigma_T(a) = \sigma_{T_2}(\sigma_{T_1}(c))$ *by definition of* σ_T, *and that* $\sigma_{T_1}(c) \in C_2$ *by definition of* σ_{T_1}. *Moreover,* $\sigma_{T_2}(\sigma_{T_1}(c)) \in C_3$ *by definition of* σ_{T_2}. *In consequence,* $a \in C_1$ *implies* $\sigma_T(a) \in C_3$.
  - *In the same way as before, we conclude that* $a \in T(C_1) \backslash C_1$ *implies* $\sigma_T(a) \in T(C_3) \backslash C_3$.
  - *For all* $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *we have that* $\sigma_{T_1}(a) \leq_{T(C_2)} \sigma_{T_1}(b)$ *by definition of* σ_{T_1}. *Moreover, we have that* $\sigma_{T_1}(a), \sigma_{T_1}(b) \in T(C_2)$ *and thus* $\sigma_{T_2}(\sigma_{T_1}(a)) \leq_{T(C_3)} \sigma_{T_2}(\sigma_{T_1}(b))$ *by definition of* σ_{T_2}. *Finally, we conclude that* $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *implies* $\sigma_T(a) \leq_{T(C_3)} \sigma_T(b)$ *by definition of* σ_T.
  - *For all* $a \in C_1|_{abstract}$ *we have that* $\sigma_{T_1}(a) \in C_2|_{abstract}$ *by definition of* σ_{T_1} *and also that* $\sigma_{T_2}(\sigma_{T_1}(a)) \in C_3|_{abstract}$ *by definition of* σ_{T_2}. *Finally,* $a \in C_1|_{abstract}$ *implies* $\sigma_T(a) \in C_3|_{abstract}$ *by definition of* σ_T.
  - *For all* $p_w \in P_1$ *we have that* $\sigma_{P_1}(p)_{\sigma_1(w)} \in P_2$ *by definition of* σ_{P_1} *and also that* $\sigma_{P_2}(\sigma_{P_1}(p))_{\sigma_2(\sigma_1(w))} \in P_3$ *by definition of* σ_{P_2}. *Finally,* $p_w \in P_1$ *implies* $\sigma_P(p)_{\sigma(w)} \in P_3$ *by definition of* σ_P.
  - *For all* $o : c \in I_1$ *we have that* $\sigma_{I_1}(o) : \sigma_{T_1}(c) \in I_2$ *by definition of* σ_{I_1} *and also that* $\sigma_{I_2}(\sigma_{I_1}(o)) : \sigma_{T_2}(\sigma_{T_1}(c)) \in I_3$ *by definition of* σ_{I_2}. *In consequence,* $o : c \in I_1$ *implies* $\sigma_I(o : c) \in I_3$.
  - *For all* $p_w(x, y) \in L_1$ *we have that* $\sigma_{P_1}(p)_{\sigma_1(w)}(\sigma_{I_1}(x), \sigma_{I_1}(y)) \in L_2$ *by definition of* σ_{I_1} *and* σ_{P_1}, *and also that* $\sigma_{P_2}(\sigma_{P_1}(p))_{\sigma_2(\sigma_1(w))}(\sigma_{I_2}(\sigma_{I_1}(x)), \sigma_{I_2}(\sigma_{I_1}(y))) \in L_3$ *by definition of* σ_{I_2} *and* σ_{P_2}. *In consequence,* $p_w(x, y) \in L_1$ *implies* $\sigma_P(p)_{\sigma(w)}(\sigma_I(x), \sigma_I(y)) \in L_3$.
- *Composition of signature morphisms is associative, i.e.* $(\sigma_3 \circ \sigma_2) \circ \sigma_1 = \sigma_3 \circ (\sigma_2 \circ \sigma_1)$*:*
  - *For each* $a \in C_1$ *we have that* $\sigma_{T_2} \circ \sigma_{T_1}(a) = \sigma_{T_2}(\sigma_{T_1}(a))$ *and thus* $\sigma_{T_3} \circ (\sigma_{T_2} \circ \sigma_{T_1})(a) = \sigma_{T_3}(\sigma_{T_2}(\sigma_{T_1}(a)))$ *by the definition of composition. Finally, this last result is equals to* $\sigma_{T_3} \circ \sigma_{T_2}(\sigma_{T_1}(a))$ *which is equals to* $(\sigma_{T_3} \circ \sigma_{T_2}) \circ \sigma_{T_1}(c)$.
  - *The proof is the same in the case of* σ_R, σ_P, *and* σ_I.
- *There exists an identity signature morphism* $id_{\Sigma_1} : \Sigma_1 \to \Sigma_1$ *defined as a tuple* $\langle id_T, id_R, id_P, id_I \rangle$ *such that* $id_T(c) = c$, $id_R(c) = c$, $id_P(c) = c$, *and* $id_I(c) = c$. *This morphism satisfies the signature morphism conditions:*
  - $a \in C_1$ *implies* $id_T(a) \in C_1$,
  - $a \in T(C_1) \backslash C_1$ *implies* $id_T(a) \in T(C_1) \backslash C_1$,
  - $a, b \in T(C_1)$ *with* $a \leq_{T(C_1)} b$ *implies* $id_T(a) \leq_{T(C_1)} id_T(b)$,
  - $a \in C_1|_{abstract}$ *implies* $id_T(a) \in C_1|_{abstract}$,
  - $p_w \in P_1$ *implies* $id_P(p)_{id(w)} \in P_1$,
  - $o : c \in I_1$ *implies* $id_I(o) : id_T(c) \in I_1$
  - $p_w(x, y) \in L_1$ *implies* $id_P(p)_{id(w)}(id_I(x), id_I(y)) \in L_1$

Finally, signatures and signature morphisms define a category. □

Lemma 12. *There is a functor* Sen *giving a set of formulas* ψ *(object in the category* Set*) for each signature* Σ *(object in the category* Sign*), as shown in the definition of a formula, and a function* $\sigma : \text{Sen}(\Sigma_1) \to \text{Sen}(\Sigma_2)$ *(arrow in the category* Set*) translating formulas for each signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$ *(arrow in the category* Sign*), as shown in the extension of the signature morphism to formulas.*

Proof. We have to prove that Sen *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By the extension of the signature morphism to formulas, the image of an arrow $\sigma : \Sigma_1 \to \Sigma_2$ in the category Sign *is an arrow σ : Sen(Σ_1) $\to$ Sen(Σ_2) in the category* Set*Also, by the definition of formulas, the image of a signature Σ in the category* Sign *is an object* Sen(Σ) *in the category* Set*Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that Sen($\sigma_2 \circ \sigma_1$) = Sen(σ_2) $\circ$ Sen(σ_1).
Let Σ_i (i=1..4) be signatures, and let σ_i:$\Sigma_i \to \Sigma_{i+1}$ (i=1, 2) be signature morphisms. Sen(σ_2) $\circ$ Sen(σ_1) *is the canonical application of the signature morphism σ_1 to the elements in ψ_1, composed with the canonical application of the signature morphism σ_2 (by definition of signature morphism extend to formulas). Since signature morphisms can be composed (as defined in Lemma 1), this is the same as the canonical application of the composition of the signature morphism to ψ_1, i.e.* Sen($\sigma_2 \circ \sigma_1$).

(c) Let $id_{\Sigma_1} : \Sigma_1 \to \Sigma_1$ be an identity signature morphism (defined in Lemma 1). We can see that identities are preserved since, by definition, for any Σ_1-formula ψ_1, $id_{\Sigma_1}(\psi_1)$ is a Σ_1-formula such that $id(r \bullet p) = id_R(r) \bullet id_P(p) = r \bullet p$.
Finally, the functor Sen *is defined.* □

Lemma 13. *For any signatures, the Σ-interpretations and Σ-homomorphisms define a category* Mod*(Σ). The points of the category are the Σ-interpretations, its arrows are the Σ-homomorphisms.*

Proof. Let $\Sigma = (\boldsymbol{T}, \boldsymbol{P}, \boldsymbol{M})$ with $\boldsymbol{T}$ = $(T(C), \leq_{T(C)}, C|_{abstract})$, $\boldsymbol{P}$ = (R, P), and $\boldsymbol{M}$ = (I, L) be a signature, let $\mathcal{I}_i$ = $(\boldsymbol{V}^{\boldsymbol{T}}_{\boldsymbol{C}}(\boldsymbol{O}_i), \boldsymbol{A}_i, K^{\mathcal{I}}_i)$ (i=1..4) be Σ-interpretations, and let $h_i : \mathcal{I}_i \to \mathcal{I}_{i+1}$ (i=1..3) be Σ-homomorphisms, then:

- *Σ-homomorphisms can be composed. We define the composition $h_2 \circ h_1$ as a a family of maps $(h_c)_{c \in T(C)}$ with $h_c : V_{c_1} \to V_{c_3}$ such that: $h_c(v) = h_{c_2}(h_{c_1}(v))$, $v \in O_{c_1}$. We have to prove that $h_2 \circ h_1$ is a Σ-homomorphism, i.e.*
  - *$h_c(v) \in O_{c_3}$ forall $v \in O_{c_1}$. By definition of homomorphism, we have that for each $v \in O_{c_1}$, $h_{c_1}(v) \in O_{c_2}$ and that for each $w \in O_{c_2}$ $h_{c_2}(w) \in O_{c_3}$, thus $h_{c_2}(h_{c_1}(v)) = h_c(v) \in O_{c_3}$.*
  - *$h_c(v) \in V_{c_3} \backslash O_{c_3}$ forall $v \in V_{c_1} \backslash O_{c_1}$. Proceeding in the same way as before, we have that for each $v \in V_{c_1} \backslash O_{c_1}$, $h_{c_2}(h_{c_1}(v)) = h_c(v) \in V_{c_3} \backslash O_{c_3}$.*
  - *$(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_c(v_1), h_c(v_2)) \in p^{\mathcal{I}_3}$ for any $v_i \in V_{c_1}$ ($i = 1, 2$) and $p(r_1 : c_1, r_2 : c_2) \in P$. By definition of homomorphism, we have that for any $v_i \in V_{c_1}$ ($i = 1, 2$) and $p(r_1 : c_1, r_2 : c_2) \in P$, $(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_{c_1}(v_1), h_{c_2}(v_2)) \in p^{\mathcal{I}_2}$, and also for any $w_i \in V_{c_2}$ ($i = 1, 2$) and $p(r_1 : c_1, r_2 : c_2) \in P$, $(w_1, w_2) \in p_2^{\mathcal{I}}$ iff $(h_{c_2}(w_1), h_{c_2}(w_2)) \in p^{\mathcal{I}_2}$. Thus, for any $v_i \in V_{c_1}$ ($i = 1, 2$) and $p(r_1 : c_1, r_2 : c_2) \in P$, $(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(h_{c_2}(h_{c_1}(v_1)), h_{c_2}(h_{c_1}(v_2))) = (h_c(v_1), h_c(v_2)) \in p^{\mathcal{I}_3}$.*
  - *$K^{\mathcal{I}}(o_1 : c) \neq K^{\mathcal{I}}(o_2 : d)$ iff $h_{c_1}(K^{\mathcal{I}}(o_1 : c)) \neq h_{c_1}(K^{\mathcal{I}}(o_2 : d))$ iff $h_{c_2}(h_{c_1}(K^{\mathcal{I}}(o_1 : c))) \neq h_{c_2}(h_{c_1}(K^{\mathcal{I}}(o_2 : d)))$, thus $h_c(K^{\mathcal{I}}(o_1 : c)) \neq h_c(K^{\mathcal{I}}(o_2 : d))$.*
- *Composition of Σ-homomorphisms is associative, i.e., $(h_3 \circ h_2) \circ h_1 = h_3 \circ (h_2 \circ h_1)$. By definition of composition of homomorphisms, for each $v \in O_{c_1}$ we have that $(h_3 \circ h_2) \circ h_1(v) = (h_3 \circ h_2)(h_{c_1}(v)) = h_{c_3}(h_{c_2}(h_{c_1}(v))) = h_3 \circ h_{c_2}(h_{c_1}(v)) = h_3 \circ (h_2 \circ h_1)$.*

- *There exist an identity Σ-homomorphism $id_{\mathcal{I}_1} : \mathcal{I}_1 \to \mathcal{I}_1$ consisting of a family of maps $(id_c)_{c \in T(C)}$ with $id_c : V_{c_1} \to V_{c_1}$ such that: $id_c(v) = v$, $v \in O_{c_1}$. It trivially holds that $id_{\mathcal{I}_1}$ is a Σ-homomorphism since:*
  - *$id_c(v) = v \in O_{c_1}$ forall $v \in O_{c_1}$.*
  - *$id_c(v) = v \in V_{c_1} \backslash O_{c_1}$ forall $v \in V_{c_1} \backslash O_{c_1}$.*
  - *$(v_1, v_2) \in p_1^{\mathcal{I}}$ iff $(id_c(v_1), id_c(v_2)) = (v_1, v_2) \in p^{\mathcal{I}_1}$ for any $v_i \in V_{c_1}$ ($i = 1, 2$) and $p(r_1 : c_1, r_2 : c_2) \in P$.*
  - *we have that $id_c(K^{\mathcal{I}}(o : c)) = K^{\mathcal{I}}(o : c)$, thus $id_c(K^{\mathcal{I}}(o_1 : c)) \neq id_c(K^{\mathcal{I}}(o_2 : d))$ iff $K^{\mathcal{I}}(o_1 : c) \neq K^{\mathcal{I}}(o_2 : d)$.*

Finally, Σ-interpretations and Σ-homomorphisms define a category. □

Lemma 14. *The reduct of Σ-interpretations and Σ-homomorphisms is a functor* Mod*(σ) from Σ_2-interpretations to Σ_1-interpretations (and Σ_2-homomorphisms to Σ_1-homomorphisms) for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$.*

Proof. By definition, domain and codomain of the reduct of an Σ-homomorphism are the reduct of domain and codomain, respectively, of the Σ-homomorphism. We have now to prove that: (a) the reduct of a composition of two Σ-homomorphisms is the composition of the reducts of those Σ-homomorphisms, and (b) that the reduct of an identity Σ-homomorphisms is likewise an identity.

Let $\Sigma = (\boldsymbol{T}, \boldsymbol{P}, \boldsymbol{M})$ with $\boldsymbol{T} = (T(C), \leq_{T(C)}, C\,|_{abstract})$, $\boldsymbol{P} = (R, P)$, and $\boldsymbol{M} = (I, L)$ be a signature, let $\mathcal{I}_i = (\boldsymbol{V_C^T}(\boldsymbol{O}_i), \boldsymbol{A}_i, K_i^{\mathcal{I}})$ (i=1..3) be Σ-interpretations, and let $h_i : \mathcal{I}_i \to \mathcal{I}_{i+1}$ (i=1, 2) be Σ-homomorphisms.

(a) $(h_2 \circ h_1)|_\sigma = h_2|_\sigma \circ h_1|_\sigma$. By definition of reduct of homomorphisms, we have that for any $c \in T(C)$, for any $v \in V_c$, $(h_2 \circ h_1)|_\sigma$ is defined by $(h_2 \circ h_1)_{\sigma(c)}(v)$. By definition of composition of homomorphisms, this is equals to $(h_2)_{\sigma(c)}((h_1)_{\sigma(c)}(v))$. Thus, by definition of composition of homomorphisms, this is equals to $((h_2)_{\sigma(c)} \circ (h_1)_{\sigma(c)})(v)$ which is the definition of $h_2|_\sigma \circ h_1|_\sigma$.

(b) Let $id_{\mathcal{I}_2}$ be an identity Σ_2-homomorphism, then $id_{\mathcal{I}_2}|_\sigma$ is an identity Σ_1-homomorphism, since by definition of reduct of a homomorphism $id_{\mathcal{I}_2}|_\sigma$ is the Σ_1-homomorphism h_1 defined by $h_{1_c}(v) = id_{\mathcal{I}_{2_{\sigma(c)}}}(v) = v$ for any $c \in T(C)$, for any $v \in V_c$.

Finally, the reduct of Σ-interpretations and Σ-homomorphisms is a functor. □

Lemma 15. *There is a functor* Mod *giving a category* Mod*(Σ) of Σ-interpretations (object in the category* **Cat***) for each signature Σ (object in the category* Sign*), as shown in Lemma 3, and a functor* Mod*(σ) (arrow in the category* **Cat***) from Σ_2-interpretations to Σ_1-interpretations (and Σ_2-homomorphisms to Σ_1-homomorphisms) for each signature morphism $\sigma : \Sigma_1 \to \Sigma_2$ (arrow in the category* Sign*), as shown in Lemma 4.*

Proof. We have to prove that Mod *is indeed a functor, i.e.: (a) domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow, (b) composition is preserved, and (c) identities are preserved.*

(a) By Lemma 4, the image of an arrow $\sigma : \Sigma_2 \to \Sigma_1$ in the category Sign^{op} *is an arrow* $\text{Mod}(\sigma) : \text{Mod}(\Sigma_2) \to \text{Mod}(\Sigma_1)$ *in the category* **Cat**. *Also, by Lemma 3, the image of a signature Σ in the category* Sign *is an object* Mod*(Σ) in the category* **Cat**. *Thus, domain and codomain of the image of an arrow are the images of domain and codomain, respectively, of the arrow.*

(b) We have to prove that $\text{Mod}(\sigma_2 \circ \sigma_1) = \text{Mod}(\sigma_2) \circ \text{Mod}(\sigma_1)$ *for both, interpretations and homomorphisms. Let Σ_i (i=1..3) be signatures, let $\sigma_i : \Sigma_i \to \Sigma_{i+1}$ (i=1, 2) be signature morphisms, let $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A}, K^{\mathcal{I}})$ be a Σ_3-interpretation, and let h be a Σ_3-homomorphism. Then, we have to prove:*

- $\mathcal{I}|_{\sigma_2 \circ\, \sigma_1} = (\mathcal{I}|_{\sigma_2})|_{\sigma_1}$.
  By definition of reduct, $\mathcal{I}|_{\sigma_2}$ is the Σ_2-interpretation $(\boldsymbol{V_C^T}(\boldsymbol{O}|_{\sigma_2}), \boldsymbol{A}|_{\sigma_2}, K^{\mathcal{I}}|_{\sigma_2})$ such that:
  - $\boldsymbol{V_C^T}(\boldsymbol{O}|_{\sigma_2}) = (V_{\sigma_2(c)})_{c \in T(C_2)}$
  - $\boldsymbol{A}|_{\sigma_2} = \{\sigma_{2p}(p)^{\mathcal{I}} \mid p \in P_2\}$
  - $K^{\mathcal{I}}|_{\sigma_2}\ (o : c) = K^{\mathcal{I}}(\sigma_{2I}(o) : \sigma_{2T}(c))$ *for all* $o : c \in I_2$

  Then $(\mathcal{I}|_{\sigma_2})|_{\sigma_1}$ is the Σ_1-interpretation $(\boldsymbol{V_C^T}((\boldsymbol{O}|_{\sigma_2})|_{\sigma_1}), (\boldsymbol{A}|_{\sigma_2})|_{\sigma_1}, (K^{\mathcal{I}}|_{\sigma_2})|_{\sigma_1})$ such that:
  - $\boldsymbol{V_C^T}((\boldsymbol{O}|_{\sigma_2})|_{\sigma_1}) = (V_{\sigma_2(\sigma_1(c))})_{c \in T(C_1)}$
  - $(\boldsymbol{A}|_{\sigma_2})|_{\sigma_1} = \{\sigma_{2p}(\sigma_{1p}(p))^{\mathcal{I}} \mid p \in P_1\}$
  - $(K^{\mathcal{I}}|_{\sigma_2})|_{\sigma_1}$ *maps each* $o : c \in I_1$ *to an element of* $V_{\sigma_2(\sigma_1(c))}$

  and this is equal to $\mathcal{I}|_{\sigma_2 \circ\, \sigma_1}$.

– $h|_{\sigma_2 \circ \sigma_1} = (h|_{\sigma_2})|_{\sigma_1}$.
By definition of reduct, $h|_{\sigma_2}$ is defined by $h|_{\sigma_2 c}(v) = h_{\sigma_2(c)}(v)$ for any $c \in T(C_2)$, for any $v \in V_c$, and thus $(h|_{\sigma_2})|_{\sigma_1}$ is defined by $(h|_{\sigma_2})|_{\sigma_1 c}(v) = h_{\sigma_2(\sigma_1(c))}(v) = h_{\sigma_2 \circ \sigma_1(c)}(v)$ for any $c \in T(C_1)$, for any $v \in V_c$, which is equals to $h|_{\sigma_2 \circ \sigma_1}$.

(c) Let $id_\sigma : \Sigma \to \Sigma$ be an identity signature morphism (defined in Lemma 1). We have to prove that Mod(id_σ) *is an identity functor, i.e., it is composed by the identity reduct of Σ-interpretations and the identity reduct of Σ-homomorphisms.*

– *By definition of reduct, for any Σ-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A}, K^{\mathcal{I}})$, $\mathcal{I}|_{id_\sigma}$ is the Σ-interpretation $(\boldsymbol{V_C^T}(\boldsymbol{O}|_{id_\sigma}), \boldsymbol{A}|_{id_\sigma}, K^{\mathcal{I}}|_{id_\sigma})$ such that:*
  - $\boldsymbol{V_C^T}(\boldsymbol{O}|_{id_\sigma}) = (V_{id_\sigma(c)})_{c \in T(C)}$
  - $\boldsymbol{A}|_{id_\sigma} = \{id_{\sigma p}(p)^{\mathcal{I}} \mid p \in P\}$
  - $K^{\mathcal{I}}|_{id_\sigma}(o : c) = K^{\mathcal{I}}(id_{\sigma I}(o) : id_{\sigma T}(c))$ *for all* $o : c \in I$

  Finally, by the definition of id_σ, $\mathcal{I}|_{id_\sigma} = \mathcal{I}$, thus $\_|_{id_\sigma}$ is the identity reduct of Σ-interpretations.
– *By definition of reduct, given a Σ-interpretation $\mathcal{I}_1 = (\boldsymbol{V_C^T}(\boldsymbol{O}_1), \boldsymbol{A}_1, K_1^{\mathcal{I}})$, for any Σ-homomorphism $h : \mathcal{I}_1 \to \mathcal{I}_2$, the reduct $h|_{id_\sigma}$ is defined by $h|_{id_\sigma c}(v) = h_{id_\sigma(c)}(v) = h_c(v)$ for any $c \in T(C)$, for any $v \in V_c$. Now, since $\mathcal{I}|_{id_\sigma} = \mathcal{I}$, we have that $\_|_{id_\sigma}$ is the identity reduct of Σ-homomorphisms.*

Finally, the functor Mod *is defined.* □

Theorem 3 (Satisfaction Condition for MOF). *Given signatures $\Sigma_i = (\boldsymbol{T}_i, \boldsymbol{P}_i, \boldsymbol{M}_i)$ $(i = 1, 2)$ with $\boldsymbol{T}_i = (T(C_i), \leq_{T(C_i)}, C_i|_{abstract})$, $\boldsymbol{P}_i = (R_i, P_i)$, and $\boldsymbol{M}_i = (I_i, L_i)$ a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-interpretation $\mathcal{I} = (\boldsymbol{V_C^T}(\boldsymbol{O}), \boldsymbol{A}, K^{\mathcal{I}})$, and a Σ_1-formula ψ, the following satisfaction condition holds.*

$$\mathcal{I}|_\sigma \models_{\Sigma_1} \psi \text{ iff } \mathcal{I} \models_{\Sigma_2} \sigma(\psi)$$

Proof. We know that $p^{\mathcal{I}|_\sigma} = \sigma_P(p)^{\mathcal{I}}$ $\forall p(r_1 : c, r_2 : d) \in P_1$ by definition of$|_\sigma$. With this result we can deduce that $(r \bullet p)^{\mathcal{I}|_\sigma} = (\sigma_R(r), \sigma_P(p))^{\mathcal{I}}$. Moreover, for all φ of the form $\#(r \bullet p) = n$, we have that $\sigma(\varphi)$ is $\#(\sigma_R(r) \bullet \sigma_P(p)) = n$, by definition of σ. Finally, using both results we have that φ is $\#(r \bullet p) = n$ and $|S| = n \ \forall \ S \in (r \bullet p)^{\mathcal{I}|_\sigma} \Leftrightarrow \sigma(\varphi)$ is $\#(\sigma_R(r) \bullet \sigma_P(p)) = n$ and $|S| = n \ \forall \ S \in (\sigma_R(r), \sigma_P(p))^{\mathcal{I}}$. Thus, $\mathcal{I}|_\sigma \models_{\Sigma_1} \varphi \Leftrightarrow \mathcal{I} \models_{\Sigma_2} \sigma(\varphi)$. We can proceed exactly in the same way for proving the other two cases of φ.

Finally, the satisfaction condition holds. □

Theorem 4 (Satisfaction Condition for QVT). *Given signatures $\Sigma_i = \langle \Sigma_1^{\mathrm{C}_i}, \Sigma_2^{\mathrm{C}_i}, \Sigma^{\mathrm{FOL}_i} \rangle (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M} = \langle \mathcal{M}_1^{\mathrm{C}_2}, \mathcal{M}_2^{\mathrm{C}_2}, \mathcal{M}^{\mathrm{FOL}_2} \rangle$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, and a Σ_1-formula $\varphi = \langle \varphi_1^{\mathrm{C}_1}, \varphi_2^{\mathrm{C}_1}, \varphi^{\mathrm{rules}} \rangle$ with variables in $X_2|_\sigma$, the following satisfaction condition holds.*

$$\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi \text{ iff } \mathcal{M} \models_{\Sigma_2} \sigma(\varphi)$$

Proof. We will first prove some preliminary results:

1. *Given $FOL^=$ signatures Σ_i $(i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-first-order structure $\mathcal{M}$, a set of variables $X_2 = (X_2^s)_{s \in S_2}$, a $\mathcal{M}$-variable assignments μ, and a Σ_1-formula φ with variables in $X_2|_\sigma$, the following condition holds by definition of the $FOL^=$ institution: $\mathcal{M}|_\sigma, \mu|_\sigma \models_{\Sigma_1} \varphi$ iff $\mathcal{M}, \mu \models_{\Sigma_2} \sigma(\varphi)$*

2. *Given signatures $\Sigma_i (i = 1, 2)$, a signature morphism $\sigma : \Sigma_1 \to \Sigma_2$, a Σ_2-model $\mathcal{M}$ with a Σ_2-first-order structure $\mathcal{M}^{\mathrm{FOL}}$, a $\mathcal{M}$-variable assignments μ, and a pattern* Pattern $= \langle E, A, Pr \rangle$*, the following condition holds: $\mathcal{M}^{\mathrm{FOL}}, \mu \models \sigma(\mathrm{Pattern})$ iff $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models$* Pattern

Proof.

$\mathcal{M}^{\mathrm{FOL}}, \mu \models \sigma(\mathrm{Pattern})$
- *iff* $\sigma(p)_D(\mu(\sigma(x)), \mu(\sigma(y))) \in \mathcal{M}^{\mathrm{FOL}}. \forall\, rel(\sigma(p), \sigma(x), \sigma(y)) \in \sigma(A)$
  - *and* $\mathcal{M}^{\mathrm{FOL}}, \mu \models_{\mathrm{FOL}} \sigma(Pr)$ — *by def. of pattern satisf.*
- *iff* $p_D(\mu|_\sigma\,(x), \mu|_\sigma\,(y)) \in \mathcal{M}^{\mathrm{FOL}}|_\sigma\,. \forall\, rel(p, x, y) \in A$ — *by def. of* $\mathcal{M}^{\mathrm{FOL}}|_\sigma$ *and* σ
  - *and* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models_{\mathrm{FOL}} Pr$ — *by result 1*

Finally, by definition of satisfaction of a pattern, we conclude that $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models \mathrm{Pattern}$ *also holds.*

3. *Given signatures* $\Sigma_i (i = 1, 2)$*, a signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$*, a* Σ_2*-model* $\mathcal{M}$ *with a* Σ_2*-first-order structure* $\mathcal{M}^{\mathrm{FOL}}$*, a* $\mathcal{M}$*-variable assignments* μ*, and a* `when` *clause* $\mathrm{when} = \langle \mathrm{when}_c, \mathrm{when}_r \rangle$ *which is part of* φ*, the following condition holds:* $\mathcal{M}^{\mathrm{FOL}}, \mu \models \sigma(\mathrm{when})$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models \mathrm{when}$

   Proof.

   $\mathcal{M}^{\mathrm{FOL}}, \mu \models \sigma(\mathrm{when})$
   - *iff* $\mathcal{M}^{\mathrm{FOL}}, \mu \models_{\mathrm{FOL}} \sigma(\mathrm{when}_c)$
     - *and* $\mathcal{M}^{\mathrm{FOL}}, \mu[\sigma(v)] \models \sigma(r). \forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r)$ *by def. of satisfaction of a when clause*

   We also know that $\mathcal{M}^{\mathrm{FOL}}, \mu \models_{\mathrm{FOL}} \sigma(\mathrm{when}_c)$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models_{\mathrm{FOL}} \mathrm{when}_c$ *by result 1. Thus, we have to prove that:*

   $$\mathcal{M}^{\mathrm{FOL}}, \mu[\sigma(v)] \models \sigma(r). \forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r) \text{ iff } \mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma\, [v] \models r. \forall(r, v) \in \mathrm{when}_r$$

   This can be proved by induction on the length of the chain of dependencies of `when` *and* `where` *clauses which is assumed to be finite as we discussed before. This means that the base case is* $\mathrm{when}_r = \emptyset$ *in which the condition trivially holds, since also* $\sigma(\mathrm{when}_r) = \emptyset$*. The inductive hypothesis is such that* $\forall(\sigma(r), \sigma(v)) \in \sigma(\mathrm{when}_r)$ *we have that* $\mathcal{M}^{\mathrm{FOL}}, \mu[\sigma(v)] \models \sigma(r)$*. iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma\, [v] \models r$ *(and the same* $\forall(r, v) \in \mathrm{when}_r$*. Thus, the inductive thesis trivially holds from these hypothesis.*
   Finally, by definition of satisfaction of a `when` *clause, we conclude that* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models \mathrm{when}$ *also holds.*

4. *Given signatures* $\Sigma_i (i = 1, 2)$*, a signature morphism* $\sigma : \Sigma_1 \to \Sigma_2$*, a* Σ_2*-model* $\mathcal{M}$ *with a* Σ_2*-first-order structure* $\mathcal{M}^{\mathrm{FOL}}$*, a* $\mathcal{M}$*-variable assignments* μ*, and a* `where` *clause* $\mathrm{where} = \langle \mathrm{where}_c, \mathrm{where}_r \rangle$ *which is part of* φ*, the following condition holds:* $\mathcal{M}^{\mathrm{FOL}}, \mu \models \sigma(\mathrm{where})$ *iff* $\mathcal{M}^{\mathrm{FOL}}|_\sigma, \mu|_\sigma \models \mathrm{where}$

   Proof. The proof is similar to the case of a `when` *clause.*

Now, we can prove the satisfaction condition as follows:

$\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi$
- *iff* $\mathcal{M}_i^{\mathrm{C}_2}|_\sigma \models^{\mathrm{C}}_{\Sigma_i^{\mathrm{C}_1}} \varphi_i^{\mathrm{C}_1}\ (i = 1, 2)$
  - *and* $\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi^{\mathrm{rules}}$ — *by def. of satisfaction relation*
- *iff* $\mathcal{M}_i^{\mathrm{C}_2} \models^{\mathrm{C}}_{\Sigma_i^{\mathrm{C}_2}} \sigma(\varphi_i^{\mathrm{C}_1})\ (i = 1, 2)$ *by def. of* $\mathcal{I}^{\mathrm{C}}$
  - *and* $\mathcal{M} \models_{\Sigma_2} \sigma(\varphi^{\mathrm{rules}})$ — *as proved next*
- *iff* $\mathcal{M} \models_{\Sigma_2} \sigma(\varphi)$ — *by def. of satisfaction relation*

We need to prove that $\mathcal{M}|_\sigma \models_{\Sigma_1} \varphi^{\mathrm{rules}}$ *iff* $\mathcal{M} \models_{\Sigma_2} \sigma(\varphi^{\mathrm{rules}})$*, which assuming that* φ^{rules} *is of the form* $\langle \mathrm{Rules}, \mathrm{top} \rangle$*, it is the same to prove that*

$$\forall\, \mathrm{Rule}_i \in \mathrm{top}.\ \mathcal{M}^{\mathrm{FOL}_2}|_\sigma, \emptyset \models \mathrm{Rule}_i \text{ iff } \forall\, \sigma(\mathrm{Rule}_i) \in \sigma(\mathrm{top}).\ \mathcal{M}^{\mathrm{FOL}_2}, \emptyset \models \sigma(\mathrm{Rule}_i)$$

For any rule $\mathrm{Rule} = \langle \mathrm{VarSet}, \mathrm{Pattern}_i\ (i = 1, 2), \mathrm{when}, \mathrm{where} \rangle$*, we have two cases:*

1. *If* $\text{WhenVarSet} = \emptyset$, $\mathcal{M}^{\text{FOL}_2}|_\sigma, \emptyset \models \text{Rule}$ *if*

$$\begin{array}{l}\forall\, \mu^1|_\sigma\, [x_1, ..., x_n] \in\ |\text{VarSet}\backslash\text{2\_VarSet}|, \\ \quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma\, [x_1, ..., x_n] \models \text{Pattern}_1 \rightarrow \\ \quad\quad \exists\, \mu^2|_\sigma\, [y_1, ..., y_m] \in |\text{2\_VarSet}|, \\ \quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma \cup \mu^2|_\sigma \models \text{Pattern}_2 \wedge \\ \quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma \cup \mu^2|_\sigma \models \text{where}))\end{array}$$

2. *If* $\text{WhenVarSet} \neq \emptyset$, $\mathcal{M}^{\text{FOL}_2}|_\sigma, \emptyset \models \text{Rule}$ *if*

$$\begin{array}{l}\forall\, \mu^w|_\sigma\, [z_1, ..., z_o] \in\ |\text{WhenVarSet}|, \\ \quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma, \mu^w|_\sigma\, [z_1, ..., z_o] \models \text{when} \rightarrow \\ \quad\quad \forall\, \mu^1|_\sigma\, [x_1, ..., x_n] \in\ |\text{VarSet}\backslash(\text{WhenVarSet} \cup \text{2\_VarSet})|, \\ \quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma \cup \mu^w|_\sigma \models \text{Pattern}_1 \rightarrow \\ \quad\quad\quad\quad \exists\, \mu^2|_\sigma\, [y_1, ..., y_m] \in |\text{2\_VarSet}|, \\ \quad\quad\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma \cup \mu^2|_\sigma \cup \mu^w|_\sigma \models \text{Pattern}_2 \wedge \\ \quad\quad\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2}|_\sigma,\ \mu^1|_\sigma \cup \mu^2|_\sigma \cup \mu^w|_\sigma \models \text{where})))\end{array}$$

In both cases we can directly use the preliminar results, plus the definition of $\mu|_\sigma$*, to conclude that the following cases also hold:*

1. *If* $\sigma(\text{WhenVarSet}) = \emptyset$

$$\begin{array}{l}\forall\, \mu^1[x_1, ..., x_n] \in\ |\text{VarSet}\backslash\text{2\_VarSet}|, \\ \quad (\ \mathcal{M}^{\text{FOL}_2},\ \mu^1[x_1, ..., x_n] \models \sigma(\text{Pattern}_1) \rightarrow \\ \quad\quad \exists\, \mu^2[y_1, ..., y_m] \in |\text{2\_VarSet}|, \\ \quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2},\ \mu^1 \cup \mu^2 \models \sigma(\text{Pattern}_2) \wedge \\ \quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2},\ \mu^1 \cup \mu^2 \models \sigma(\text{where})))\end{array}$$

2. *If* $\sigma(\text{WhenVarSet}) \neq \emptyset$

$$\begin{array}{l}\forall\, \mu^w[z_1, ..., z_o] \in\ |\text{WhenVarSet}|, \\ \quad (\ \mathcal{M}^{\text{FOL}_2}, \mu^w[z_1, ..., z_o] \models \sigma(\text{when}) \rightarrow \\ \quad\quad \forall\, \mu^1[x_1, ..., x_n] \in\ |\text{VarSet}\backslash(\text{WhenVarSet} \cup \text{2\_VarSet})|, \\ \quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2},\ \mu^1 \cup \mu^w \models \sigma(\text{Pattern}_1) \rightarrow \\ \quad\quad\quad\quad \exists\, \mu^2[y_1, ..., y_m] \in |\text{2\_VarSet}|, \\ \quad\quad\quad\quad\quad (\ \mathcal{M}^{\text{FOL}_2},\ \mu^1 \cup \mu^2 \cup \mu^w \models \sigma(\text{Pattern}_2) \wedge \\ \quad\quad\quad\quad\quad\quad \mathcal{M}^{\text{FOL}_2},\ \mu^1 \cup \mu^2 \cup \mu^w \models \sigma(\text{where}))))\end{array}$$

Finally, we conclude that $\forall\, \sigma(\text{Rule}_i) \in \sigma(\text{top}).\ \mathcal{M}^{\text{FOL}_2}, \emptyset \models \sigma(\text{Rule}_i)$*. Note that we can also read this demonstration downside up, thus the satisfaction condition holds.* □