Correct-by-construction model composition Application to the Invasive Software Composition method

Mounira Kezadri Hamiaz
Université de Toulouse, IRIT, France
mounira.kezadri@enseeiht.fr

Benoît Combemale
Université de Rennes 1, IRISA, France
benoit.combemale@irisa.fr

Marc Pantel
Université de Toulouse, IRIT, France
marc.pantel@enseeiht.fr

Xavier Thirioux
Université de Toulouse, IRIT, France
xavier.thirioux@enseeiht.fr

Composition technologies improve reuse in the development of large-scale complex systems. Safety critical systems require intensive validation and verification activities. These activities should be compositional in order to reduce the amount of residual verification activities that must be conducted on the composite in addition to the ones conducted on each components. In order to ensure the correctness of compositional verification and assess the minimality of the residual verification, the contribution proposes to use formal specification and verification at the composition operator level. A first experiment was conducted in [14] using proof assistants to formalize the generic composition technology ISC and prove that type checking was compositional. This contribution extends our early work to handle full model conformance and study the mandatory residual verification. It shows that ISC operators are not fully compositional with respect to conformance and provides the minimal preconditions on the operators mandatory to ensure compositional conformance. The appropriate operators from ISC (especially bind) have been implemented in the COQ4MDE framework that provides a full implementation of MOF in the COQ proof assistant. Expected properties, respectively residual verification, are expressed as post, respectfully pre, conditions for the composition operators. The correctness of the compositional verification is proven in COQ.

1 Introduction

Composition technologies improve reuse in the development of large-scale complex systems. Safety critical systems require intensive validation and verification activities. These activities should be compositional in order to reduce the amount of residual activities that must be conducted on the composite in addition to the ones conducted on each components. In order to ensure the correctness of compositional verification and assess the minimality of the residual verification, the contribution proposes to use formal specification and verification at the composition operator level.

A first experiment was conducted in [14] using proof assistants to formalize the generic composition technology ISC [1] and especially the bind and extend operators. This generic composition method enables to enrich the models to express composition interfaces and to assemble the generated components using some composition operators. Type checking for models based on metamodels was proved to be compositional for these operators. However, the implementation of operators in ISC does not take into account other semantics properties for the conformance relation for metamodels and inconsistent models can be generated.

This contribution extends our early work to handle full model conformance and study the mandatory residual verification. It shows that ISC operators are not fully compositional with respect to conformance and provides the minimal preconditions on the operators mandatory to ensure compositional

Submitted to:

conformance. The appropriate operators from ISC (especially bind) have been implemented in the COQ4MDE framework that provides a full implementation of MOF in the COQ proof assistant. Expected properties, respectively residual verification, are expressed as post, respectively pre, conditions for the composition operators. The correctness of the compositional verification is proven in COQ.

This paper focuses on an evolution of the ISC operators (especially the bind operator) to correct the inconsistencies in the first implementation that allowed to build model compositions that do not conform to the composite metamodel. It also gives the verification for some generic semantics properties of the MOF metamodel conformance relation [18].

This first section has given a short introduction. The second section presents the required notions about COQ4MDE. To motivate the evolution of the ISC operators and the associated proofs, the third section gives an example of an inconsistent metamodel assembled by the REUSEWARE[1] [12] [11] plugin from consistent metamodels. The fourth section first discusses the formalization of the bind operator, then presents some preconditions for the conformance verification and the associated proofs. The fifth section presents some related work. Finally, the last section concludes and gives some perspectives.

2 Coq4MDE

This section gives the main insight of our MDE framework COQ4MDE, derived from [25]. It defines principally the notions of Model and MetaModel.

In our framework, the concept of metamodel is not a specialization of the concept of model. A model is the instance level and a metamodel is a modeling language used to define models. Both are formally defined in the following way. Let us consider two sets of labels: Classes, respectively References, represents the set of all possible class, respectively reference labels. Then, let us consider instances of such classes, the set Objects of object labels. References includes a specific *inh* label used to specify the inheritance relation. In the next sections, we will elide the word label and directly talk about classes, references and objects.

Definition 1 (Model) *Let $\mathscr{C} \subseteq$ Classes be a set of classes.*
Let $\mathscr{R} \subseteq \{\langle c_1, r, c_2\rangle \mid c_1, c_2 \in \mathscr{C}, r \in$ References$\}$[2] be a set of references between classes.

A Model over $\mathscr{C}$ and $\mathscr{R}$, written $\langle MV, ME\rangle \in Model(\mathscr{C},\mathscr{R})$ is a multigraph built over a finite set MV of typed object vertices and a finite set ME[3] of reference edges such that:

$$MV \subseteq \{\langle o, c\rangle \mid o \in \texttt{Objects}, c \in \mathscr{C}\}$$
$$ME \subseteq \left\{ \langle\langle o_1, c_1\rangle, r, \langle o_2, c_2\rangle\rangle \;\middle|\; \langle o_1, c_1\rangle, \langle o_2, c_2\rangle \in MV, \langle c_1, r, c_2\rangle \in \mathscr{R} \right\}$$

Note that, in case of inheritance, the same object label will be used several times in the same model graph. It will be associated to the different classes in the inheritance hierarchy going from a root to the class used to create the object. This label reuse encodes the inheritance polymorphism, a key aspect of most OO languages. Inheritance is represented in the metamodel with a special reference called *inh*. The subClass property is presented in the Section 4.

Definition 2 (Metamodel) *A MetaModel is a multigraph representing classes as vertices and references as edges as well as semantic properties over instantiation of classes and references. It is represented as*

[1] http://www.reuseware.org
[2] $\langle c_1, c_2, r\rangle$ in the COQ code is denoted here for simplification as: $\langle c_1, r, c_2\rangle$.
[3] $\langle\langle o_1, c_1\rangle, r, \langle o_2, c_2\rangle\rangle$ is denoted in the COQ code as: $\langle\langle o_1, c_1\rangle, \langle o_2, c_2\rangle, r\rangle\rangle$.

a pair composed of a multigraph (MMV, MME) *built over a finite set* MMV *of vertices and a finite set* MME *of edges, and of a predicate over models representing the semantic properties.*

A `MetaModel` *is a pair* $\langle (MMV, MME), conformsTo \rangle$ *such that:*

$$MMV \subseteq Classes$$
$$MME \subseteq \{\langle c_1, r, c_2 \rangle \mid c_1, c_2 \in MMV, r \in References\}$$
$$conformsTo : Model(MMV, MME) \rightarrow Bool$$

Given one `Model` M and one `MetaModel` MM, we can check conformance using the $conformsTo$ predicate embedded in MM. It identifies the set of valid models with respect to a metamodel.

In a prospect to construct a formal framework for model composition, we extend the previous `MDE` framework to formalize and prove the properties preservation for the `ISC` basic composition operators implemented inside the REUSEWARE framework.

3 An example of inconsistent metamodel generated by REUSEWARE

`ISC` is a generic technology for extending a DSML with model composition facilities. Its first version was defined to compose `Java` programs and was implemented in the COMPOST system[4]. A universal extension called U-ISC was proposed in [11], this technique deals with textual components that can be described using context-free grammars and then the fragments are represented as trees. The method as presented considers tree merging for the composition. Recently, in order to deal with graphical languages the method was extended to support typed graphs in [12], this method was implemented in the REUSEWARE framework. This last implementation is consistent with the description of models as graphs in our COQ4MDE framework.

Using the REUSEWARE plugin, the composition of the two models presented in Figures 1 and 2 by the composition program presented in Figure 3 generates the model presented in Figure 4.

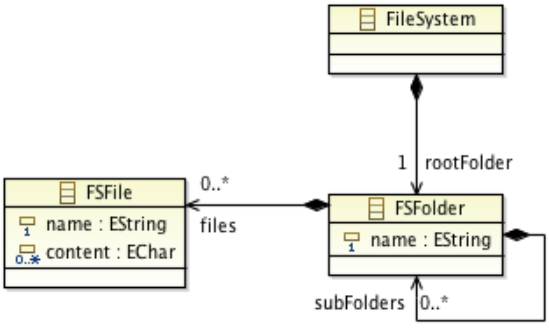

Figure 1: The advice model

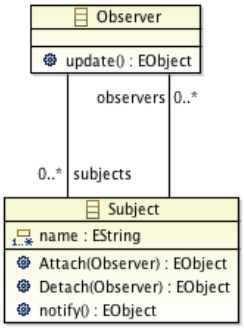

Figure 2: The observer model

This example is presented in [13] and is accessible with the REUSEWARE Eclipse plugin applications[5]. We slightly modified the observer model by adding an attribute `name` having as minimal multiplicity 1 and as maximal multiplicity $*$ (see Figure 2) to illustrate the issue with the result of the composition (see Figure 4).

[4] http://www.the-compost-system.org

[5] `http://www.reuseware.org/index.php/Reuseware_Aspect_Weaving`

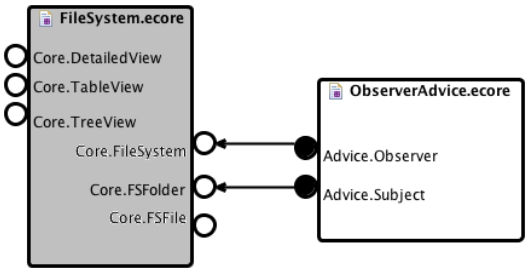

Figure 3: The composition program

The composition program shown in Figure 3 describes the links between the variation and reference points and aims to implement the class weaving for the two metamodels.

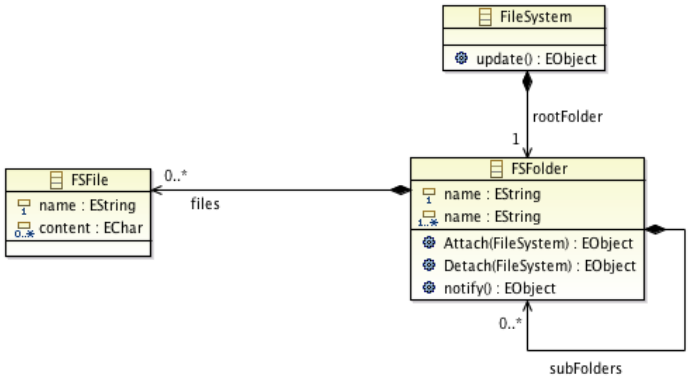

Figure 4: The composition result

In the model obtained by composition, the class FSFolder has two attributes `name` with different multiplicities (1 and 1 .. *). This generates ambiguities and the metamodel is clearly inconsistent.

Our approach for the verification allows to detect and avoid this kind of inconsistencies by considering the metamodel semantics properties. The fact that one attribute must have a single value for the minimum and maximum multiplicities is a semantics property represented with the attributes *lower* and *upper* for the class *Property* (see Figure 5)

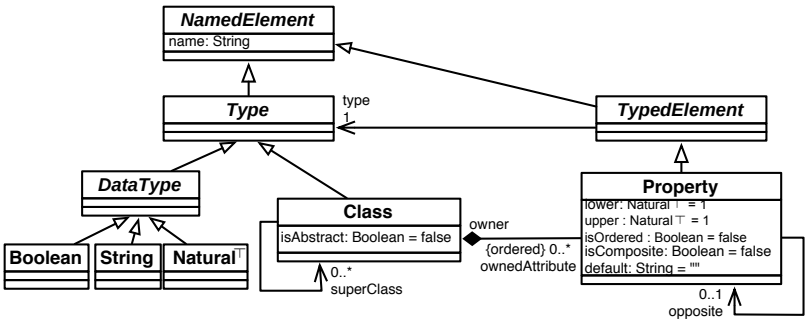

Figure 5: The `MOF` metamodel

We consider our metamodels as models conforming to `MOF` (represented in Figure 5 as a metamodel), then we verify the conformance properties in relation with this metamodel. We show in the following the verifications of this kind of properties for the `ISC` method `bind` operator.

4 The verifications

The bind operator formalized in [14] enables for two models M and M' to replace a model's element b from the model M referenced by a variation point by another model's element $\mathtt{b}'$ from the model M' referenced by a reference point. The two model's elements b and b' must have the same type. This operator as presented in [14] is proved compositional for typing but can generate inconsistencies on the resulted models with respect to conformance. The predicate InstanceOf is used to check that all objects and links of a model are instance of classes and references in a metamodel.

$$InstanceOf(\langle\langle \mathtt{MV}, \mathtt{ME}\rangle, \langle\langle \mathtt{MMV}, \mathtt{MME}, conformsTo\rangle\rangle) \triangleq$$
$$\forall\ \langle \mathtt{o}, \mathtt{c}\rangle \in \mathtt{MV}, \mathtt{c} \in \mathtt{MMV} \wedge\ \forall\langle\langle \mathtt{o}, \mathtt{c}\rangle, \mathtt{r}, \langle \mathtt{o}', \mathtt{c}'\rangle\rangle \in \mathtt{ME},\ \langle \mathtt{c}, \mathtt{r}, \mathtt{c}'\rangle \in \mathtt{MME}$$

Then, this predicate is used to verify that using two components instance of MM, the component resulting from the application of the bind operator is also instance of MM.

Let consider now the inheritance property represented using the relation *superClass* in Figure 5. This property is formally represented in COQ4MDE with a special reference called *inh*. The property subClass states that c_2 is a direct subclass of c_1 in the model $\langle MV, ME\rangle$.

$$subClass(c_1, c_2 \in \mathtt{Classes}, \langle MV, ME\rangle) \triangleq \forall o\ \in \mathtt{Objects}, \langle o : c_2\rangle \in MV \Rightarrow \langle\langle o : c_2\rangle, inh, \langle o : c_1\rangle\rangle \in ME$$

In Figure 6, we show that the bind operator generates inconsistencies concerning the inheritance. In this example, we apply on the model the bind operator with as parameters $(c : C)$ and $(c' : C')$, so that replaces the model's element $(c : C)$ by $(c' : C')$. The condition for this operator is that C is equal to C', this preserves the type safety but generates problems with the inheritance. The cause is that this replacement does not preserve the label reuse used to implement the inheritance and discussed in the Section 2.

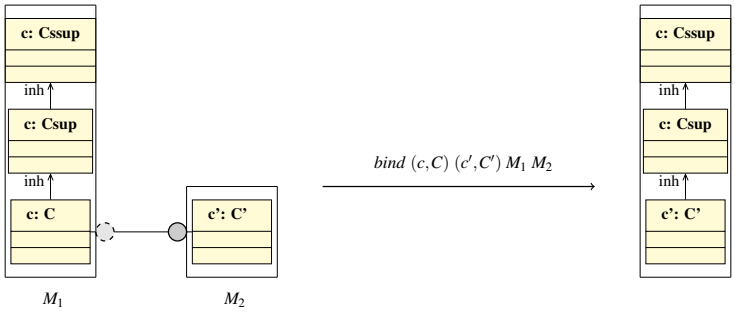

Figure 6: Inconsistency (concerning the inheritance) generated by the bind operator

To correct this inconsistency, we slightly modified this operator. The new operator changes the name of all elements named c by c', the type of each element remains unchanged. We prove then the preservation of the type safety, the inheritance and other MOF properties by this operator. The bind operator is also extended to a recursive form to support several variation and reference points as mentioned in Figure 7.

We reuse *InstanceOf* predicate to prove the type safety for the new *bind* operator using the theorem 7[6] (ValidBind) for any two models $\mathtt{M}_1$ and $\mathtt{M}_2$ and any models' elements $\mathtt{o}_1$ and $\mathtt{o}_2$.

Theorem 1 (ValidBind)
$InstanceOf\ (\mathtt{M}_1, \mathtt{MM}) \wedge InstanceOf\ (\mathtt{M}_2, \mathtt{MM}) \rightarrow InstanceOf\ ((bind\ \mathtt{o}_1\ \mathtt{o}_2\ \mathtt{M}_1\ \mathtt{M}_2), \mathtt{MM})$

[6] http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#ValidBind

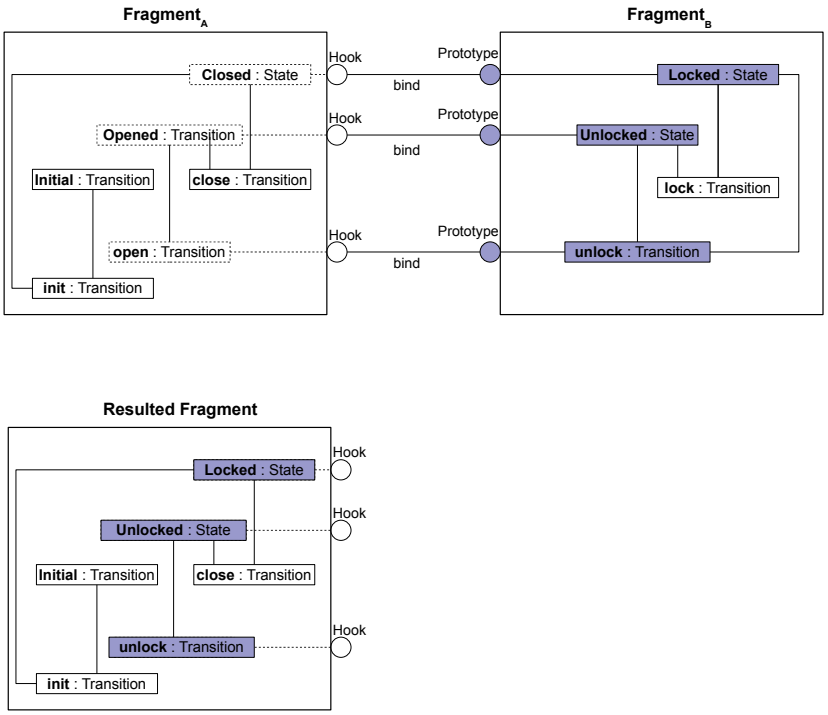

Figure 7: The `bind` of several variation points operator

PROOF SKETCH: We suppose that the two models M_1 and M_2 are instance of the metamodel MM and we prove that the model obtained by applying the *bind* operator on the two models using any two model's elements o_1 and o_2 is also instance of the metamodel MM. We verify first that o_1 is a $Hook$ for the model M_1 and o_2 is a $Prototype$ for the model M_2 and that o_1 and o_2 have the same type otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a $Hook$ and o_2 is a $Prototype$ and the two model's elements have the same types, we show that the *bind* does not change the types of the vertices and edges and so preserves the type safety (some details of the proof are given as an appendix).

We developed an elegant way to prove that the basic composition operators preserve the conformance regarding the semantics properties of metamodel (other than typing). We used this method to take into account some semantics properties of the `MOF` metametamodel. This approach prevents us trying to extract the properties from the initial metamodel which is not obvious to do. The complexity is linked to the fact that the `conformsTo` predicate is defined in a generic way to support any kind of properties on the metamodel. The idea is to ensure that each elementary property verified on the initial models is also verified on the result of the application of the composition operator. So, if the initial models are conform to some metamodel, the resulting model is consistent with the same metamodel. The basic semantics properties considered are: inheritance (`subClass`), abstract classes (`isAbstract`), multiplicities (`lower` and `upper`), the opposite references (`isOpposite`) and composite references (`areComposite`).

In follows, we present for each property, the theorem that proves the preservation for the `bind` operator and the link to the complete COQ proof.

4.1 The verification of some `MOF` properties

We show that inheritance, abstract classes, multiplicities, opposite and composite references are preserved by the `bind` operator. We note also that the proofs of these properties require in some cases

additional preconditions that represent the residual verification activities when composing verified models.

The subClass property: The theorem 2 (`BindSubClassPreserved`) shows[7] that the inheritance is preserved by the `bind` operator. So, for all classes c_1 and c_2 and for all model's elements o_1 and o_2, if c_1 is a `subClass` of c_2 in two models M_1 and M_2, then c_1 is a `subClass` of c_2 in the resulting model from (`bind` o_1 o_2 M_1 M_2).

Theorem 2 (BindSubClassPreserved)
$\forall\, \mathtt{M_1}\ \mathtt{M_2} \in \mathit{Model},\ \mathtt{c_1}\ \mathtt{c_2} \in \mathit{Classes},\ \mathtt{o_1}\ \mathtt{o_2} \in \mathit{Objects},$
$(\mathit{subClass}\ \mathtt{c_1}\ \mathtt{c_2}\ \mathtt{M_1}) \wedge\ (\mathit{subClass}\ \mathtt{c_1}\ \mathtt{c_2}\ \mathtt{M_2}) \rightarrow \mathit{subClass}\ \mathtt{c_1}\ \mathtt{c_2}\ (\mathit{bind}\ \mathtt{o_1}\ \mathtt{o_2}\ \mathtt{M_1}\ \mathtt{M_2})$

PROOF SKETCH: We prove that any two classes c_1 and c_2 linked by the *inh* relation in any two models M_1 and M_2, are also linked by the *inh* relation in the model obtained from the *bind* of any two model's elements o_1 and o_2 using the two models M_1 and M_2. We suppose that we have a relation *inh* between any two model's elements typed by c_1 and c_2 in the two models M_1 and M_2. We verify first that o_1 is a *Hook* for the model M_1 and o_2 is a *Prototype* for the model M_2 and that o_1 and o_2 have the same type otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a *Hook* and o_2 is a *Prototype* and the two model's elements have the same type, we show that the *bind* does not change the types of model's elements and the types of relations and so in the resulted model we have always an *inh* relation between any model's elements typed by c_1 and c_2. The COQ proof is long but straightforward and considers all the cases of equality between the name of any model's element typed by c_1 and the names of the model's elements o_1 and o_2 and shows in all cases that the *inh* relation is preserved.

So, there is no necessary precondition on the parameters of the `bind` operator to verify that the `subClass` property is compositional.

The isAbstract property: Abstract classes that are specified in a metamodel using the *isAbstract* attribute are not suitable for instantiation. They are often used to represent abstract concepts or entities.

$$\mathit{isAsbstract}(c_1 \in \mathtt{Classes}, \langle MV, ME\rangle) \triangleq$$
$$\forall\, o \in \mathtt{Objects}, \langle o : c_1\rangle \in MV \Rightarrow \exists\, c_2 \in \mathtt{Classes}, \langle\langle o : c_2\rangle, \mathit{inh}, \langle o : c_1\rangle\rangle \in ME$$

The preservation of this property by the `bind` operator is proved[8] using the theorem 3. This theorem shows that all the abstract classes in any two models $\mathtt{M_1}$ and $\mathtt{M_2}$ are also abstract in the model obtained by the application of the `bind` operator on the two models.

Theorem 3 (BindIsAbstractPreserved)
$\forall\, \mathtt{M_1}\ \mathtt{M_2} \in \mathit{Model},\ \mathtt{c} \in \mathit{Classes},\ \mathtt{o_1}\ \mathtt{o_2} \in \mathit{Objects},$
$(\mathit{isAbstract}\ \mathtt{c}\ \mathtt{M_1}) \wedge\ (\mathit{isAbstract}\ \mathtt{c}\ \mathtt{M_2}) \rightarrow \mathit{isAbstract}\ \mathtt{c}\ (\mathit{bind}\ \mathtt{o_1}\ \mathtt{o_2}\ \mathtt{M_1}\ \mathtt{M_2})$

PROOF SKETCH: We prove that any abstract class c in any two models M_1 and M_2, is also abstract in the model obtained from the *bind* of any two model's elements o_1 and o_2 using the two models M_1 and M_2. We suppose that the class c is abstract in the models M_1 and M_2. We verify first that o_1 is a *Hook* for the model M_1 and o_2 is a *Prototype* for the model M_2 and that o_1 and o_2 have the same types otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a *Hook* and o_2 is a *Prototype* and the two model's elements have the same type, we show that the *bind* does not change the types of model's

[7] `http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MSCP`
[8] `http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MIAP`

elements and so in the resulted model if an element typed by the class c is in the resulting model, then another element having the same name typed by c_1 and linked to the first model's element with an *inh* relation will be also in the resulting model. The COQ proof is long and considers all the cases of equality between the name of any model's element typed by c and the names of the model's elements o_1 and o_2 and shows in all cases that the relation is preserved.

So, there is no precondition on the parameters of the `bind` operator to verify that the `isAbstract` property is compositional.

The lower & upper properties: A minimum and maximum number of instances of target attribute or reference can be defined using the *lower* and *upper* attributes. Both attributes are used to represent a range of possible numbers of instances. Unbounded ranges can be modelled using the $\top$ value for the *upper* attribute.

$$\begin{aligned} &\mathit{lower}(\mathtt{c_1} \in \mathtt{MMV}, \mathtt{r_1} \in \mathtt{MME}, \mathtt{n} \in \mathit{Natural}^{\top}) \triangleq \langle \mathtt{MV}, \mathtt{ME} \rangle \mapsto \\ &\quad \forall\, o \in \mathtt{Objects}, \langle \mathtt{o} : \mathtt{c_1} \rangle \in \mathtt{MV} \Rightarrow |\{\mathtt{m_2} \in \mathtt{MV} \mid \langle \langle \mathtt{o} : \mathtt{c_1} \rangle, \mathtt{r_1}, \mathtt{m_2} \rangle \in \mathtt{ME}\}| \geq \mathtt{n} \end{aligned}$$

The theorem 4 (`BindLowerPreserved`) shows[9] that the `lower` property is preserved by the `bind` operator. The verification requires the `bind` operator to be injective and preserves the difference between the elements in the resulting model. This is ensured if the model's element $\mathtt{o_2}$ is not in the first model, this verifies that the `bind` operator does not add an element that already exists in the model. Finally, the preservation of the `lower` property is proven. An analogous formalization for the `lower` property is defined for the `upper` property replacing $\geq$ by $\leq$.

Theorem 4 (BindLowerPreserved)
$\forall\, \langle \mathtt{MV_1}, \mathtt{ME_1} \rangle\ \langle \mathtt{MV_2}, \mathtt{ME_2} \rangle \in \mathit{Model},\ \mathtt{c} \in \mathit{Classes},\ \mathtt{r} \in \mathit{References},$
$\mathtt{n} \in \mathit{Natural}^{\top},\ (\mathtt{o_1} : \mathtt{c_1})\ (\mathtt{o_2} : \mathtt{c_2}) \in \mathit{Objects},$
$\mathtt{c_1} = \mathtt{c_2} \wedge\ (\forall\, c, (\mathtt{o_2} : \mathtt{c}) \notin \mathtt{MV_1}) \wedge\ \mathit{Injectif\ bind}$
$\wedge\ (\mathit{lower}\ \mathtt{c}\ \mathtt{r}\ \mathtt{n}\ \langle \mathtt{MV_1}, \mathtt{ME_1} \rangle) \wedge\ (\mathit{lower}\ \mathtt{c}\ \mathtt{r}\ \mathtt{n}\ \langle \mathtt{MV_2}, \mathtt{ME_2} \rangle)$
$\rightarrow (\mathit{lower}\ \mathtt{c}\ \mathtt{r}\ \mathtt{n}\ (\mathit{bind}\ (\mathtt{o_1} : \mathtt{c_1})\ (\mathtt{o_2} : \mathtt{c_2})\ \langle \mathtt{MV_1}, \mathtt{ME_1} \rangle\ \langle \mathtt{MV_2}, \mathtt{ME_2} \rangle)).$

PROOF SKETCH: We suppose for any two models $\langle \mathtt{MV_1}, \mathtt{ME_1} \rangle$ and $\langle \mathtt{MV_2}, \mathtt{ME_2} \rangle$ that a lower bound n is satisfied for the class c in relation with the reference r (maximum n model's elements are related by the relation r to the same instance of the class c). Then, we prove that this lower bound n is also satisfied in the model obtained from the *bind* of any two model's elements o_1 and o_2 using the two models $\langle \mathtt{MV_1}, \mathtt{ME_1} \rangle$ and $\langle \mathtt{MV_2}, \mathtt{ME_2} \rangle$. We verify first like in the previous proofs that o_1 is a *Hook* for the model $\langle \mathtt{MV_1}, \mathtt{ME_1} \rangle$ and o_2 is a *Prototype* for the model $\langle \mathtt{MV_2}, \mathtt{ME_2} \rangle$ and that o_1 and o_2 have the same types otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a *Hook* and o_2 is a *Prototype* and the two model's elements have the same types, we show that the *bind* does not change the types of the model's elements and does not reduce the lower bound in the resulting model because the *bind* is supposed injective and so does not introduce new model's elements duplications. The COQ proof is long and uses intermediate lemmas to simplify iterations and calculations of the links and the model's elements (the difficulty is linked to the elegant coding of the graphs and the models using dependent types). This proof considers also all the cases of equality between the name of any instance of c and the names of the model's elements o_1 and o_2 and shows in all cases that the lower bound is preserved.

[9] `http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MLP`

The preservation of the upper property is described[10] by the BindUpperPreserved theorem which is similar to the previous theorem for the lower property. So, we find it necessary to introduce assumptions about the model's elements to ensure that the composition using the bind operator preserves the lower and upper properties. There are therefore preconditions on the bind operator to ensure the preservation of these properties.

The isOpposite property: A reference can be associated to an *opposite* reference. It implies that, in a valid model, for each link instance of this reference between two objects, a link in the opposite direction between the same objects exists.

$$isOpposite(\mathtt{r_1}, \mathtt{r_2} \in \mathtt{MME}) \triangleq \langle \mathtt{MV}, \mathtt{ME} \rangle \mapsto \forall\, \mathtt{m_1}, \mathtt{m_2} \in \mathtt{MV}, \langle \mathtt{m_1}, \mathtt{r_1}, \mathtt{m_2} \rangle \in \mathtt{ME} \Leftrightarrow \langle \mathtt{m_2}, \mathtt{r_2}, \mathtt{m_1} \rangle \in \mathtt{ME}$$

The theorem 5 (BindIsOppositePreserved) shows[11] that each pair of opposite references in the two models M_1 and M_2 are also opposite in the resulting model from applying the bind operator on the two models. Finally, the property isOpposite is preserved.

Theorem 5 (BindIsOppositePreserved)
$\forall\, \mathtt{M_1}\ \mathtt{M_2} \in \mathit{Model},\ \mathtt{r_1}\ \mathtt{r_2} \in \mathit{References},\ \mathtt{o_1}\ \mathtt{o_2} \in \mathit{Objects},$
$(\mathit{isOpposite}\ \mathtt{r_1}\ \mathtt{r_2}\ \mathtt{M_1}) \wedge\ (\mathit{isOpposite}\ \mathtt{r_1}\ \mathtt{r_2}\ \mathtt{M_2}) \rightarrow (\mathit{isOpposite}\ \mathtt{r_1}\ \mathtt{r_2}\ (\mathit{bind}\ \mathtt{o_1}\ \mathtt{o_2}\ \mathtt{M_1}\ \mathtt{M_2})).$

PROOF SKETCH: We prove that any two references r_1 and r_2 that are opposite in any two models M_1 and M_2, are also opposite in the model obtained from the *bind* of any two model's elements o_1 and o_2 using the two models M_1 and M_2. We verify first like in all the other proofs that o_1 is a *Hook* for the model M_1 and o_2 is a *Prototype* for the model M_2 and that o_1 and o_2 have the same type otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a *Hook* and o_2 is a *Prototype* and the two model's elements have the same type, we show that the *bind* does not change the references and so we can find all the opposite references from the initial models. The COQ proof considers all the cases of equality between the names of the model's elements and the names of o_1 and o_2 and shows in all cases the preservation of the opposite references.

So, there is no precondition on the parameters of the bind operator to verify that the isOpposite property is compositional.

The areComposite property: A reference can be *composite* and, as a matter of fact, defining a set of references considered as a whole to be composite, instead of a single one, appears closer to the intended meaning. In such a case, instances of the target concept belong to a single instance of source concepts.

$$\begin{aligned} &areComposite(\mathtt{c_1} \in \mathtt{MMV}, R \subseteq \mathtt{MME}) \triangleq \langle \mathtt{MV}, \mathtt{ME} \rangle \mapsto \\ &\quad \forall\, o \in \mathtt{Objects} \Rightarrow |\{\mathtt{m_1} \in \mathtt{MV} \mid \langle \mathtt{m_1}, \mathtt{r}, \langle \mathtt{o} : \mathtt{c_1} \rangle \rangle \in \mathtt{ME}, \mathtt{r} \in R\}| \leq 1 \end{aligned}$$

The theorem 6 (BindAreCompositeSubsPreserved) shows[12] that the set of composite references in the two models $\mathtt{M_1}$ and $\mathtt{M_2}$ are also composite in the resulted model from the application of the bind operator on the two models. This theorem requires also that the bind operator is injective and requires that the substituted model does not contain an element whose name is $\mathtt{o_2}$.

[10] http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MUP
[11] http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MIOP
[12] http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html#Bind2MACP

Theorem 6 (BindAreCompositeSubsPreserved)
$\forall\ \langle \mathtt{MV}_1, \mathtt{ME}_1\rangle\ \langle \mathtt{MV}_2, \mathtt{ME}_2\rangle \in \mathit{Model},\ \mathtt{c} \in \mathit{Classes},$
$\mathtt{R} \subset \mathit{References},\ \mathtt{o}_1\ \mathtt{o}_2 \in \mathit{Objects},\ \mathtt{c}_2\ \mathtt{c}_2 \in \mathit{Classes},$
$\mathtt{c}_1 = \mathtt{c}_2\ \wedge\ (\forall\, c, (\mathtt{o}_2 : \mathtt{c}) \notin \mathtt{MV}_1) \wedge$ *Injectif bind*
$\wedge\ (\mathit{areComposite}\ \mathtt{c}\ \mathtt{R}\ \langle \mathtt{MV}_1, \mathtt{ME}_1\rangle) \wedge\ (\mathit{areComposite}\ \mathtt{c}\ \mathtt{R}\ \langle \mathtt{MV}_2, \mathtt{ME}_2\rangle)$
$\rightarrow (\mathit{areComposite}\ \mathtt{c}\ \mathtt{R}\ (\mathit{bind}\ (\mathtt{o}_1 : \mathtt{c}_1)\ (\mathtt{o}_2 : \mathtt{c}_2)\ \langle \mathtt{MV}_1, \mathtt{ME}_1\rangle\ \langle \mathtt{MV}_2, \mathtt{ME}_2\rangle))$

PROOF SKETCH: We suppose for any two models $\langle \mathtt{MV}_1, \mathtt{ME}_1\rangle$ and $\langle \mathtt{MV}_2, \mathtt{ME}_2\rangle$, for any instance of a class c in these models, at most one ancestor is linked with a composite reference. We verify that this property is also satisfied in the model obtained from the *bind* of any two model's elements o_1 and o_2 using these two models. We verify first like in the previous proofs that o_1 is a *Hook* for the model $\langle \mathtt{MV}_1, \mathtt{ME}_1\rangle$ and o_2 is a *Prototype* for the model $\langle \mathtt{MV}_2, \mathtt{ME}_2\rangle$ and that o_1 and o_2 have the same type otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a *Hook* and o_2 is a *Prototype* and the two model's elements have the same type, we show that the *bind* does not change the types of the model's elements and does not increase the number of composite references for any model's element and this by supposing like for the *lower* property proof that the *bind* is injective and so does not introduce new model's elements duplications. The COQ proof is long and uses intermediate lemmas to simplify iterations and calculations of the references and models' elements (the difficulty is linked to elegant coding of graphs and models using dependent types). The proof considers also all the cases of equality between the name any instance of c in the two models and the names of the model's elements o_1 and o_2 and shows in all cases that the limit for the number of composite relation for any model's element is preserved.

4.2 The `bind` operator with several variation points

This version is a generalization of the `bind` operator. It is characterized by a list l of variation and reference points (`bind` of two Models with Several Hooks).

$\mathtt{Bind2MSH} : \mathit{Model} \times \mathit{Model} \times \mathit{list}\ (\mathit{Objects} \times \mathit{Objects})$ is defined as:

$$\mathtt{Bind2MSH}\ \mathtt{M}_1\ \mathtt{M}_2\ l = \forall\ (o, o') \in l, \mathit{bind}\ \mathtt{M}_1\ \mathtt{M}_2\ o\ o'\ l$$

The proofs of properties require the following assumptions: type compatibility between two model's elements for each pair of elements in the list, the `bind` operator to be injective and an additional condition: the same `Prototype` is not given more than once to ensure the preservation of multiplicities. The same assumptions/preconditions are necessary to prove the compositional verification of the various considered properties.

The proofs for this version of the `bind` operator use the proofs of the `bind` basic operator in addition to a standard schema to find the target model and the application conditions. The language of tactics for the COQ system [9] is used to define the tactics that significantly improved the proofs[13].

4.3 The `extend` operator

Two variations of the `extend` operator are implemented. The first version makes the hypothesis in addition to the `extend` operator definition that the two models are disjoint to define the predicate *extensibleC*.

The second version does not make any assumptions about the intersection of models. In this latest version, models can contain common elements as they may result from the extraction of components

[13] `http://coq4mde.enseeiht.fr/FormalMDE/Bind2M_Verif.html`

from the same model. We do not present the proofs of the MOF properties for these operators in this paper, but the proofs are finalized and the interested reader can refer to the special page[14].

This work presents the preconditions allowing for each operator (ISC basic operators) to generate consistent metamodels. Detecting and resolving the conflicts require the compositional application of several composition operators (each operator is proved preserving the properties) and contributes for the satisfaction of the next applied operator preconditions. For example, this can be used to find a sensible unification of the constraints contributed by the two model's fragments being composed.

5 Related work

In the first version of the ISC composition method [1], the notion of conformance is restricted to the *instanceOf* property defined in [14]. A composition operator is safe if it preserves the consistence (Theorem 5.1 (Sound Composition retains Consistency) in [1]). The first version of ISC was defined on fragments of Java code, the extension operator guarantees by definition that it will not change the code of a fragment box, although it can change its semantics. The semantics is preserved if the added code to the variation point is independent of the code of the fragment box (Theorem 5.2 (Sound Composition with Extension Composers) in [1]). We proved that the semantics is preserved if the models are disjoint (Section 4.3). Moreover, we extended this work by offering the preconditions that enable to preserve the semantics and all the formal proofs in the COQ proof assistant.

In the last version of ISC [13] implemented in the REUSEWARE framework as an Eclipse plugin and developed in parallel with our work, the typing property is ensured in relation with some properties of the MOF metamodel using the notion of compatibility between the variation and reference points. But and as presented in the motivating example of this paper, this version does not take into account all the semantics properties of the MOF metamodel and inconsistent metamodels can be generated by composition. We presented then the theorems proving the preservations of some of the MOF semantics properties and the preconditions for the verification.

Our approach is original compared to the work of Aßman [1], we provide in advance the preconditions which ensure that the result of applying an operator is valid (typing and semantics properties). We do not need to check for each application that the result is valid, but we know the preconditions that must be met and if our conditions are satisfied, we can ensure that the result of the composition is consistent. The expected direct consequences for our work are: the use of COQ4MDE to prove the correction of the ISC method itself and the composition methods in general by introducing and proving the preconditions ensuring the properties preservation.

This work is also closely related to all work about the formalization of model driven engineering, we present first in what follows some approaches based on shallow encoding and then compare them to our formalization. We present finally briefly a deep encoding for the MDE concepts associated with a highlight for the differences with our encoding.

MoMENT (MOdel manageMENT) [4] is an algebraic model management framework that provides a set of generic operators to manipulate models. In the MoMENT framework, the metamodels are represented as algebraic specifications and the operators are defined independently of the metamodel using the MAUDE language [8]. To be used, the operators must be specified in a module called signature that specifies the constructs of the metamodel. The approach was implemented in a tool[15] that gives also an automatic translation from an EMF metamodel to a signature model.

[14] http://coq4mde.enseeiht.fr/FormalMDE/Extend_Verif.html

[15] http://moment.dsic.upv.es/

A. Vallecillo et al. have designed and implemented a different embedding of metamodels, models ([24]) and model transformations ([26]) using MAUDE. This embedding relies on the object rewriting semantics in order to implement model transformations.

I. Poernomo has proposed an encoding of metamodels and models using type theory ([21]) in order to allow correct by construction development of model transformation using proof-assistants like COQ ([22]). Some simple experiments have been conducted using COQ mainly on tree-shaped models ([23]) using inductive types. General `graph` model structure can be encoded using co-inductive types. However, as shown in [20] by C. Picard and R. Matthes, the encoding is quite complex as COQ enforces structural constraints when combining inductive and co-inductive types that forbid the use of the most natural encodings proposed by Poernomo et al. M. Giorgino et al. rely in [10] on a spanning tree of the graph combined with additional links to overcome that constraint using the ISABELLE proof-assistant. This allows to develop a model transformation relying on slightly adapted inductive proofs and then extract classical imperative implementations.

The HOL-OCL system [5] [6] is an environment for interactive modelling with `UML` and `OCL` that can be used for example to prove class invariants.

These embeddings are all shallow: they rely on sophisticated similar data structure to represent model elements and metamodels (e.g. COQ (co-)inductive data types for model elements and object and (co-)inductive types for metamodel elements).

The work described in this paper is a deep embedding, each concept from models and metamodels was encoded in [25] using elementary constructs instead of relying on similar elements in MAUDE, COQ or ISABELLE. The purpose of this contribution is not to implement model transformation using correct-by-construction tools but to give a kind of denotational semantics for model-driven engineering concepts that should provide a deeper understanding and allow the formal validation of the various implemented technologies. Other work aiming to define a semantics for a modelling language by explicitly and denotationally define the kind of systems the language describes and to focus on the variations and variability in the semantics [7] [17]. Compared to the last work, we are interested in a complete and unique formalisation of the conformity to metamodels, of course this property must be considered in the more general consistency relation and we are focused mainly in the proof of the preservation of this conformity relation by the `ISC` composition operators.

Another formalisation in COQ of the `MDE` concepts by F.Barbier et al is accessible[16] [2], this representation is attached to the proof of the properties shown in [15] (on instantiation relations and model transformations). The last formalization differs from ours by a detailed representation of the different components of models and metamodels based on the `MOF` concepts. The COQ4MDE formalisation has the advantage to be more generic and minimum through the use of modules for the representation of these concepts and its support for a large variety of properties describing the conformity by a predicate integrated to the metamodel type.

6 Conclusion

We have addressed the problem of compositional verification for models relying on the generic composition method `ISC` and the REUSEWARE framework. We first proposed in [14] a formalization for the `ISC` composition method and the verification of type safety for these operators. Then, we presented in this paper the verification of generic semantics properties in relation with the `MOF` metametamodel.

[16] `http://web.univ-pau.fr/~barbier/Coq/`

This integration enables to extract executable correct by construction composition operators. The termination of the extracted operators is ensured by the COQ definition. The typing property and a set of semantics properties in relation with the MOF metametamodel are proved preserved directly or by the composition operators by introducing some preconditions on the parameters of the composition operators. The application is not limited to a specific language, but can be extended to all models and modeling languages defined by metamodels. From the ISC composition method basic operators (bind and extend), more complex operators were built. The complex operators allow more complex transformations such as linking several variation points at the same time.

This proposal is a required step in the formalization of compositional verification techniques. The next step of our work is to formalize other composition operators and to take into account others static constraints such as OCL constraints [19] and more dynamic properties such as the deadlock freedom proposed in the BIP framework [3]. The expected result of our work is to define a correct by construction framework for combining several composition techniques.

References

[1] U. Aßmann. *Invasive software composition*. Springer, 2003.

[2] F. Barbier, P. Castéran, E. Cariou, and O. Le Goaer. Adaptive Software based on Correct-by-Construction Metamodels. In B. C. P. G.-B. O. S. M. V. Garcia Diaz, J.M. Cueva Lovelle, editor, *Progressions and Innovations in Model-Driven Software Engineering*, Advances in Systems Analysis, Software Engineering, and High Performance Computing (ASASEHPC), pages 308–325. IGI Global, July 2013.

[3] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *Software Engineering and Formal Methods, 2006. SEFM 2006. Fourth IEEE International Conference on*, pages 3–12. IEEE, 2006.

[4] A. Boronat and J. Meseguer. An algebraic semantics for MOF. *Formal Aspects of Computing*, 22(3-4):269–296, 2010.

[5] A. D. Brucker and B. Wolff. A proposal for a formal ocl semantics in isabelle/hol. In *Theorem Proving in Higher Order Logics*, pages 99–114. Springer, 2002.

[6] A. D. Brucker and B. Wolff. Hol-ocl: a formal proof environment for uml/ocl. In *Fundamental Approaches to Software Engineering*, pages 97–100. Springer, 2008.

[7] M. V. Cengarle, H. Grönniger, B. Rumpe, and M. Schindler. System model semantics of class diagrams. *Technische Universitat Braunschweig*, 2008.

[8] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. Maude: specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.

[9] D. Delahaye. A tactic language for the system Coq. In *Logic for Programming and Automated Reasoning*, pages 377–440. Springer, 2000.

[10] M. Giorgino, M. Strecker, R. Matthes, and M. Pantel. Verification of the Schorr-Waite algorithm–from trees to graphs. In *Logic-Based Program Synthesis and Transformation*, pages 67–83. Springer, 2011.

[11] F. Heidenreich, J. Henriksson, J. Johannes, and S. Zschaler. On Language-Independent Model Modularisation. In S. Katz, H. Ossher, R. France, and J.-M. Jézéquel, editors, *Transactions on Aspect-Oriented Software Development VI*, volume 5560 of *Lecture Notes in Computer Science*, pages 39–82. Springer Berlin Heidelberg, 2009.

[12] J. Henriksson, F. Heidenreich, J. Johannes, S. Zschaler, and U. Aßmann. Extending grammars and metamodels for reuse: the Reuseware approach. *IET software*, 2(3):165–184, 2008.

[13] J. Johannes. *Component-based model-driven software development.* PhD thesis, Dresden University of Technology, 2010.

[14] M. Kezadri, B. Combemale, M. Pantel, X. Thirioux, et al. A proof assistant based formalization of MDE components. In *8th International Symposium on Formal Aspects of Component Software (FACS 2011)*, 2011.

[15] T. Kühne. Matters of (meta-) modeling. *Software & Systems Modeling*, 5(4):369–385, 2006.

[16] L. Lamport. How to write a proof. *The American mathematical monthly*, 102(7):600–608, 1995.

[17] S. Maoz, J. O. Ringert, and B. Rumpe. Semantically configurable consistency analysis for class and object diagrams. In *Model Driven Engineering Languages and Systems*, pages 153–167. Springer, 2011.

[18] OMG. OMG Meta Object Facility (MOF) Core Specification (Version 2.4.1). *Available on: http://www.omg.org/spec/MOF/2.4.1*, 2.4.1, 2011.

[19] OMG. OMG Object Constraint Language (OCL), Version 2.3.1, January 2012.

[20] C. Picard and R. Matthes. Coinductive graph representation : the problem of embedded lists. *Electronic Communications of the EASST, Special issue Graph Computation Models, GCM'10*, 2011.

[21] I. Poernomo. The meta-object facility typed. In *SAC*, pages 1845–1849, 2006.

[22] I. Poernomo. Proofs-as-model-transformations. In *ICMT*, pages 214–228, 2008.

[23] I. Poernomo and J. Terrell. Correct-by-construction model transformations from partially ordered specifications in Coq. In *ICFEM*, pages 56–73, 2010.

[24] J. R. Romero, J. E. Rivera, F. Durán, and A. Vallecillo. Formal and tool support for Model Driven Engineering with Maude. *Journal of Object Technology*, 6(9):187–207, 2007.

[25] X. Thirioux, B. Combemale, X. Crégut, and P. Garoche. A Framework to Formalise the MDE Foundations. In R. Paige and J. Bézivin, editors, *International Workshop on Towers of Models (TOWERS)*, pages 14–30, Zurich, June 2007.

[26] J. Troya and A. Vallecillo. Towards a rewriting logic semantics for ATL. In *ICMT*, pages 230–244, 2010.

A A part from the ValidBind theorem proof

This appendix presents the mathematical proof for the first theorem presented in the Section 4. The theorem *ValidBind* proves the preservation of the $instanceOf$ property by the `bind` operator. The Lamport's method [16] is used to write this proof.

Theorem 7 (ValidBind)
$InstanceOf\ (\mathtt{M_1},\mathtt{MM}) \wedge InstanceOf\ (\mathtt{M_2},\mathtt{MM}) \rightarrow InstanceOf\ ((bind\ \mathtt{o_1}\ \mathtt{o_2}\ \mathtt{M_1}\ \mathtt{M_2}),\mathtt{MM})$

ASSUME: M_1 the $\mathtt{Model}\langle \mathtt{MV},\mathtt{ME}\rangle$
M_2 the $\mathtt{Model}\langle \mathtt{MV_1},\mathtt{ME_1}\rangle$
MM the $\mathtt{MetaModel}\langle \mathtt{MMV},\mathtt{MME},conformsTo\rangle$
H: $InstanceOf\ (\langle \mathtt{MV},\mathtt{ME}\rangle,\langle \mathtt{MMV},\mathtt{MME},conformsTo\rangle)$.
H_{M2}: $InstanceOf\ (\langle \mathtt{MV_1},\mathtt{ME_1}\rangle,\langle \mathtt{MMV},\mathtt{MME},conformsTo\rangle)$.
PROVE: $InstanceOf\ ((bind\ \mathtt{o_1}\ \mathtt{o_2}\ \langle \mathtt{MV},\mathtt{ME}\rangle\ \langle \mathtt{MV_1},\mathtt{ME_1}\rangle),\langle \mathtt{MMV},\mathtt{MME},conformsTo\rangle)$
PROOF SKETCH: We suppose that the two models M_1 and M_2 are instance of the metamodel MM and we prove that the model obtained by applying the *bind* operator on the two models using two model's elements o_1 and o_2 is also instance of the metamodel MM. We verify first that o_1 is a $Hook$ for the model M_1 and o_2 is a $Prototype$ for the model M_2 and that o_1 and o_2 have the same types otherwise the *bind* returns the model M_1 and the proof is trivial. In case of o_1 is a $Hook$ and o_2 is a $Prototype$ and the two model's elements have the same type (the case detailed below), we show that the *bind* does not change the types of the vertices and edges and so preserves the type safety.
PROOF:
$\langle 1\rangle$1. After introducing the definitions of $instanceOf$ and the *bind* operator, the hypothesis H becomes:
H: $(\forall\langle \mathtt{o},\mathtt{c}\rangle,\langle \mathtt{o},\mathtt{c}\rangle \in \mathtt{MV} \rightarrow \mathtt{c} \in \mathtt{MMV})$
$\wedge\ (\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o'},\mathtt{c'}\rangle\rangle,\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o'},\mathtt{c'}\rangle\rangle \in \mathtt{ME} \rightarrow \langle \mathtt{c},\mathtt{r},\mathtt{c'}\rangle \in \mathtt{MME})$.

The current goal is transformed:
$(\forall\langle \mathtt{o},\mathtt{c}\rangle, \langle \mathtt{o},\mathtt{c}\rangle \in (V.image\ mapv\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g) \rightarrow \mathtt{c} \in \mathtt{MMV})$
$\wedge (\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle, \langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle \in (E.image\ mapv\ mapa\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g)$
$\rightarrow \langle \mathtt{c},\mathtt{r},\mathtt{c}'\rangle \in \mathtt{MME})$.

$\langle 1\rangle 2$. The hypothesis H is divided into two hypotheses:
H_0: $\forall\langle \mathtt{o},\mathtt{c}\rangle, \langle \mathtt{o},\mathtt{c}\rangle \in \mathtt{MV} \rightarrow \mathtt{c} \in \mathtt{MMV}$.
H_1: $\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle, \langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle \in \mathtt{ME} \rightarrow \langle \mathtt{c},\mathtt{r},\mathtt{c}'\rangle \in \mathtt{MME}$.
The current goal is divided into two sub-goals:
1. $\langle \mathtt{o},\mathtt{c}\rangle \in (V.image\ mapv\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g) \rightarrow \mathtt{c} \in \mathtt{MMV}$
2. $\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle, \langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle \in (E.image\ mapv\ mapa\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g)$
$\rightarrow \langle \mathtt{c},\mathtt{r},\mathtt{c}'\rangle \in \mathtt{MME}$

$\langle 2\rangle 1$. We begin by proving the first subgoal that corresponds to the left side of the conjunction:
ASSUME: H_2: $\langle \mathtt{o},\mathtt{c}\rangle \in (V.image\ mapv\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g)$.
PROVE: $(\mathtt{c} \in \mathtt{MMV})$
PROOF:
$\langle 3\rangle 1$. By generalizing the lemma 1 using H_2, we get a new hypothesis: H_4: $\exists(o',c') \in MV \mid mapv\ (o',c') = (o,c)$.
$\langle 3\rangle 2$. We introduce the definition of *mapv*, we can conclude that $c' = c$
then we have as hypothesis: $H_5 : (o',c) \in MV$.
$\langle 3\rangle 3$. By applying H_0 with as parameter (o',c) and H_5.
$\langle 3\rangle 4$. Q.E.D.

$\langle 2\rangle 2$. We now prove the second part of the goal:
$\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle, \langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle \in (E.image\ mapv\ mapa\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g)$
$\rightarrow \langle \mathtt{c},\mathtt{r},\mathtt{c}'\rangle \in \mathtt{MME}$
ASSUME: Having as an additional hypothesis to H_0 and H_1, the hypothesis H_2: $\forall\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle$, $\langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle \in (E.image\ mapv\ mapa\ (\langle \mathtt{MV},\mathtt{ME}\rangle)\ g)$
PROVE: This sub-goal can be resolved by proving: $\langle \mathtt{c},\mathtt{r},\mathtt{c}'\rangle \in \mathtt{MME}$
PROOF:
$\langle 3\rangle 1$. Here, we generalize the lemma 2 using the hypothesis H_2, we get a new hypothesis:
H_4: $\exists\langle\langle \mathtt{o}_1,\mathtt{c}_1\rangle,\mathtt{r}_1,\langle \mathtt{o}'_1,\mathtt{c}'_1\rangle\rangle$,
$\langle\langle \mathtt{o}_1,\mathtt{c}_1\rangle,\mathtt{r}_1,\langle \mathtt{o}'_1,\mathtt{c}'_1\rangle\rangle \in \mathtt{ME} \mid mape\ \langle\langle \mathtt{o}_1,\mathtt{c}_1\rangle,\mathtt{r}_1,\langle \mathtt{o}'_1,\mathtt{c}'_1\rangle\rangle = \langle\langle \mathtt{o},\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}',\mathtt{c}'\rangle\rangle$.
$\langle 3\rangle 2$. We introduce the definition of *mape*, we can conclude that:
$c_1 = c$, $c'_1 = c'$ et $r_1 = r$,
then we have as hypothesis: $H_5 : \langle\langle \mathtt{o}_1,\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}'_1,\mathtt{c}'\rangle\rangle \in \mathtt{ME}$.
$\langle 3\rangle 3$. By applying H_0 with as parameter $\langle\langle \mathtt{o}_1,\mathtt{c}\rangle,\mathtt{r},\langle \mathtt{o}'_1,\mathtt{c}'\rangle\rangle$ and H_5.
$\langle 3\rangle 4$. Q.E.D.

□

Lemmas used in this proof are:

Lemma 1 (V.imageElim)
$\forall\ mapv,\ mapa,\ \langle \mathtt{MV},\mathtt{ME}\rangle \in Model, v \in (image\ \langle \mathtt{MV},\mathtt{ME}\rangle) \rightarrow \exists w \in MV \wedge mapv\ w = v.$

Lemma 2 (E.imageElim)
$\forall\ mapv,\ mapa,\ \langle \mathtt{MV},\mathtt{ME}\rangle \in Model, e \in (image\ \langle \mathtt{MV},\mathtt{ME}\rangle) \rightarrow \exists w \in \mathtt{ME} \wedge mape\ w = e.$

The proofs of these two lemmas are constructed by induction on the structure of the graph and involve other theorems that are not presented here but are available with our COQ code.