FRAGMENTA: A Theory of Fragmentation for MDE

Nuno Amálio
University of York (UK)

Juan de Lara and Esther Guerra
Universidad Autónoma de Madrid (Spain)

***Abstract*—Model-Driven Engineering (MDE) promotes models throughout development. However, models may become large and unwieldy even for small to medium-sized systems. This paper tackles the MDE challenges of model complexity and scalability. It proposes FRAGMENTA, a theory of modular design that breaks down overall models into fragments that can be put together to build meaningful wholes, in contrast to classical MDE approaches that are essentially monolithic. The theory is based on an algebraic description of *models*, *fragments* and *clusters* based on graphs and morphisms. The paper's novelties include: (i) a mathematical treatment of fragments and a seaming mechanism of *proxies* to enable inter-fragment referencing, (ii) *fragmentation strategies*, which prescribe a fragmentation structure to model instances, (iii) FRAGMENTA's support for both top-down and bottom-up design, and (iv) our formally proved result that shows that inheritance hierarchies remain well-formed (acyclic) globally when fragments are composed provided some local fragment constraints are met.**

***Index Terms*—Model-driven engineering, meta-modelling, modularity, graphs, scalability, model composition**

I. INTRODUCTION

The construction of large software systems entails issues of complexity and scalability. Model-Driven Engineering (MDE) emphasises design; it raises the level of abstraction by promoting models to primary artifacts of software development. The goal is to master and alleviate the complexity of software through abstraction; despite abstraction, models' sizes can still be overwhelmingly large and complex even for small to medium-size systems, impairing comprehensibility and complicating the refinement of models into running systems.

This paper presents FRAGMENTA, a mathematical theory that tackles the complexity and scalability challenges of modern day MDE. FRAGMENTA is based on the ideas of *modularity* and *separation of concerns* [1], allowing an overall model to be broken down into *fragments* that are organised around *clusters*. A fragment is a smaller model, a sub-model of an ensemble constituting the overall model. FRAGMENTA is a modular approach that supports both top-down and bottom-up ways of building bigger fragments from smaller ones that covers both the instance and type perspectives of models (also known as models and metamodels). The theory presented here is based on *proxies*, which act as the *seams* or *joints* of fragments and enable inter-fragment referencing, mimicking a similar mechanism of the popular EMF [2].

FRAGMENTA's primary goal is to establish a mathematical theory of model fragmentation for MDE that is formally verified and validated, and that provides a sound and rigorous foundation for FRAGMENTA implementations as part of MDE environments, languages, frameworks and tools. The theory is built upon the algebraic theory of graphs and their morphisms. Its complexity was tackled with the aid of formal languages and tools, namely: the Z language and its CZT typechecker, and the Isabelle proof assistant [3]. All formal proofs that validate and verify the theory were undertaken in Isabelle.

Contributions. The paper's contributions are as follows:

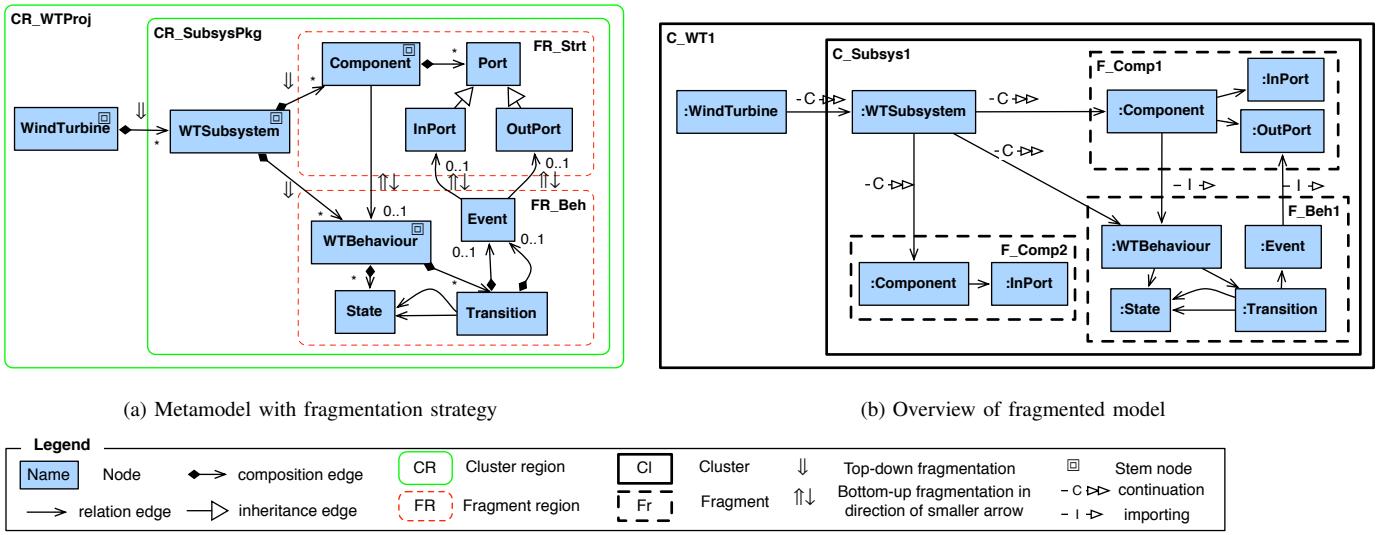
- A mathematical theory of model fragments and the associated seaming mechanism of proxies, which mimics a similar mechanism used in practice [2]. To our knowledge, this particular combination together with a study on the particularities of proxies, is missing in similar works.
- A formal treatment of the meta-level notion of fragmentation strategies, which is, to our knowledge, missing in other theories such as ours.
- The formally proved result that demonstrates that our local fragment constraints ensure that the resulting compositions will be inheritance cycle free, a fundamental well-formedness property of object-oriented inheritance, precluding the need for global checks.
- A theory of incremental definition based on proxies that supports both bottom-up and top-down design. The top-down concept of continuation is novel, as far as we know.
- FRAGMENTA's three-level architecture: local fragment, global fragment and cluster, which is, to our knowledge, absent in previous works.

Paper Outline. The remainder of this paper starts by giving an overview of FRAGMENTA (sec. II). Then, it presents: FRAGMENTA's graph-based foundations (sec. III), the basis of fragmented graphs and the way they are organised and clustered (sec. IV), the way to compose fragmented models to obtain monolithic models (sec. V), and FRAGMENTA's notion of typing and fragmentation strategies (sec. VI). The paper then concludes by discussing the presented results (sec. VII) and their relation to related work (sec. VIII), and by briefly summarising the paper's main findings (sec. IX) .

II. FRAGMENTA IN A NUTSHELL

FRAGMENTA is a theory to design fragmented models. It enables the construction of model fragments that can be processed and understood in isolation and put together to make consistent and meaningful bigger fragments. An overall model is a collection of fragments. FRAGMENTA's primitive units are *fragments*, *clusters* and *models*:

- A fragment is a graph with *proxy* nodes for referencing that act as *seams* or *joints*; proxies are surrogates that represent some other element of some fragment.
- Clusters are containers to put related fragments together. They provide means for hierarchical organisation: a cluster may contain other clusters and fragments.



(a) Metamodel with fragmentation strategy

(b) Overview of fragmented model

Fig. 1: Running Example: metamodel with fragmentation strategy and fragmented model instance

- A model is a collection of fragments organised with clusters. This enables fragmentations that mimic modern programming projects; in implementations, fragments may be deployed as files and clusters as folders.

FRAGMENTA supports both top-down and bottom-up fragmented designs based on *imports* and *continuations*. Fragmentation strategies (FSs) are metamodel annotations that stipulate a fragmentation structure to model instances.

Figure 1 presents our running example, based on an industrial language to model software controllers for wind turbines (WTs) taken from the MONDO EU project[1]. Figure 1a shows this language's metamodel, and Fig. 1b presents an abstracted instance model that omits proxy nodes. WT controllers are organised in subsystems made up of components, containing several input and output ports. A component's behaviour is described by a state machine. The metamodel's FS defines regions (rounded rectangles) of type cluster (solid line) or fragment (dashed line). Related instances of the nodes inside a region must pertain to a corresponding instance-level cluster or fragment. Hence, this FS stipulates the following:

- WT models are placed in clusters (region `CR_WTProj`), containing clusters for each WT subsystem (region `CR_SubsysPkg`), which contain a structural and a behavioural fragment (regions `FR_strt` and `FR_beh`, respectively). A region's *stem* node (symbol ⊡) indicates that the creation of its instances entails the creation of corresponding instance-level cluster or fragment.
- A FS specifies how cross-border associations are to be fragmented. We consider two alternatives: *top-down* (symbol ⇓) and *bottom-up* (symbol ⇑). Top-down fragmentations are realised as *continuations*; bottom-up as *importings*. In Fig. 1a, cross-border edges coming out of `WindTurbine` and `WTSubsystem` are top-down; the remaining ones, bottom-up.

Continuations and imports constitute alternative ways to use proxies. If a fragment `B` imports or continues a fragment `A`, it means, in both cases, that `B` may have proxies that reference elements of `A`. The difference lies in the way in which the relation between the two fragments is interpreted, leading to two different ways of making use of incremental definition. A fragment to be continued is *extended* by its continuing fragments; this gives top-down because it is like defining the root of a tree that is continued through proxies in the leafs (continuations). Importing gives bottom-up, because the tree grows by defining composite fragments that import leaf fragments and that contain proxies to elements in the leafs.

The overview model of Fig. 1b (a detailed model is given in Fig. 6) complies with its metamodel FS.

III. GRAPHS AS THE FOUNDATIONS OF FRAGMENTA

FRAGMENTA's foundations lie on graphs and their morphisms. We present most notions informally and intuitively. Most formal definitions are given in the appendix, an abridged version of the accompanying technical report [4].

We assume sets V and E of all possible nodes and edges of graphs (def. 1). As usual, a graph G, a member of set Gr (def. 2), is made of sets $V_G \subseteq V$ and $E_G \subseteq E$ of nodes and edges, and (total) functions $s,t\colon E_G \to V_G$ for the source and target of edges (see Fig. 2a). Graph morphisms (G-morphisms, def. 3) are made of two functions mapping nodes and edges, and preserving the source and target functions – functions fV and fE depicted in Fig. 2b. G-morphisms can be composed (def. 4). Graphs and their morphisms form category **Graph**.

Structural graphs (SGs) enrich graphs to support MDE *conceptual* or *structural* models, like UML class diagrams. Typically, such models include: (i) *inheritance* relations, (ii)

[1]http://www.mondo-project.org/

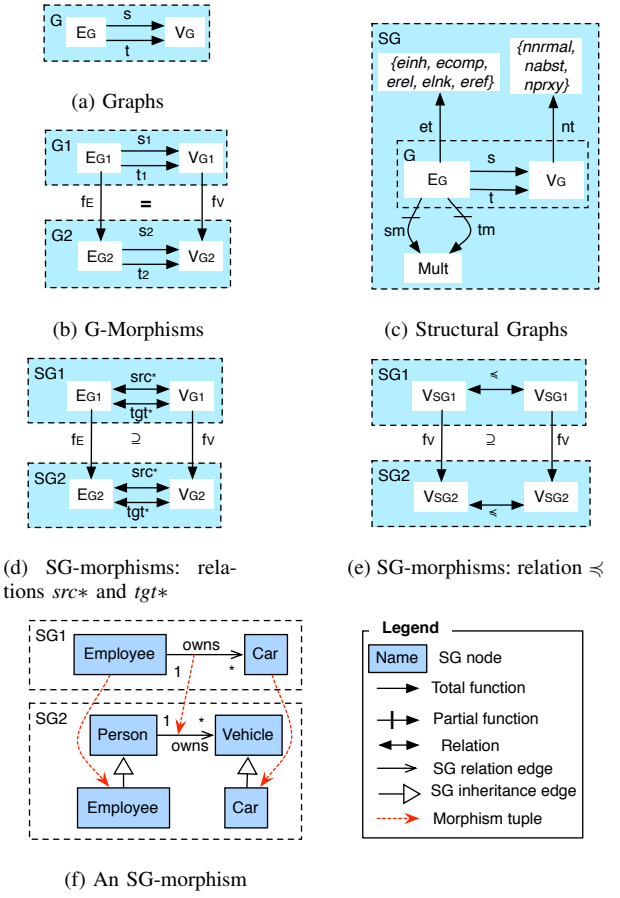


Fig. 2: Graphs, graph morphisms, structural graphs

containment, *whole-part* or *composition* relations, and (iii) relations subject to multiplicity constraints.

An SG, member of set *SGr* (def. 7), is a tuple $SG = (G, nt, et, sm, tm)$ (see Fig. 2c), comprising: (a) a graph $G : Gr$, (b) two colouring functions *nt*, *et* giving the kinds of nodes and edges, and (c) two partial multiplicity functions *sm*, *tm* to assign multiplicities to the source and target of edges.

SGs (Fig. 2c) support edges of type inheritance (*einh*), composition (*ecomp*), relation (*erel*), link (*elnk*) and reference (*eref*, used by proxies in sec. IV-A). We call *association* edges to edges of type composition, relation and link. All relation and composition edges (and no other) have multiplicities. Inheritance is reified with edges, and we permit dummy self edges (to enable more morphisms), but require the inheritance graph formed by restricting to non-self inheritance edges to be acyclic. SGs' node types are normal (*nnrml*), abstract (*nabst* for abstract classes) and proxy (*nprxy*). Fig. 2f shows two SGs.

SG-morphisms (def. 8) cater to the semantics of inheritance: the association edges of parent nodes become edges of child nodes. In Fig. 2f, `owns` of `SG2` is also an edge of nodes `Employee` and `Car`. To capture this semantics, we introduce functions src^* and tgt^*, which yield relations $E \leftrightarrow V$ between edges and vertices that extend functions *s* and *t* to support the fact that an edge can have more than one source or target

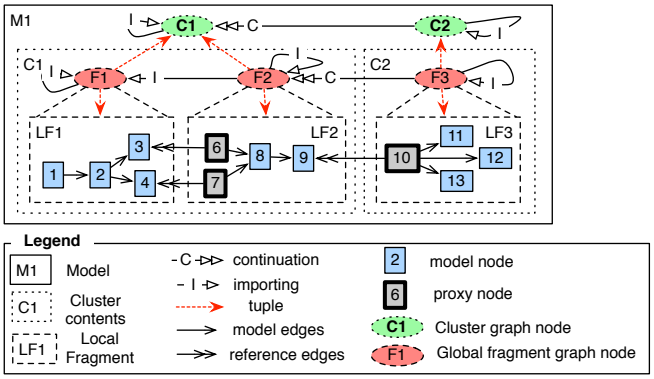


Fig. 3: A model (`M1`) made up of two clusters (`C1` and `C2`) and three fragments (`F1`, `F2` and `F3`). A model has three levels: cluster (C_i), global fragment (F_i) and local fragment (LF_i).

node (see [4])[2]. The transition from G- to SG-morphisms considers this new set-up: the equality commuting expressed in terms of functional composition (Fig. 2b) is replaced by subset commuting expressed in terms of relation composition (Fig. 2d). Likewise, for the actual inheritance relation between nodes, captured by relation ≼; SG morphisms may shrink (removing nodes) or extend (adding nodes) inheritance hierarchies and they should, therefore, preserve the inheritance information, which is described as subset commuting (Fig. 2e). SG-morphisms disregard the preservation of multiplicities and colouring (considered as part of typing, sec. VI). SGs and their morphisms form category **SGraphs** (see [4]).

Figure 2f presents a valid SG-morphism. It is also possible to build a (non-injective) morphism from `SG2` to `SG1` by adding dummy inheritance self-edges to `SG1` (omitted in figures); both morphisms were proved correct in Isabelle [4].

IV. Fragmented Models

Figure 3 sketches a FRAGMENTA model, comprising two clusters and three fragments. It highlights FRAGMENTA's three-layered architecture, local fragment (LF_i), global fragment (F_i) and cluster (C_i), related through morphisms.

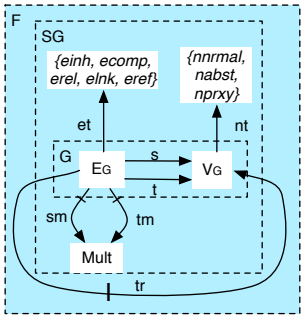
A. Fragments

Fragments provide referencing, allowing proxies to refer to other nodes, possibly belonging to other fragments. This is realised through reference edges (introduced as part of SGs in sec. III), which point to themselves in SGs – they are *unreferenced*[3]. Fragments provide the actual targets of reference edges.

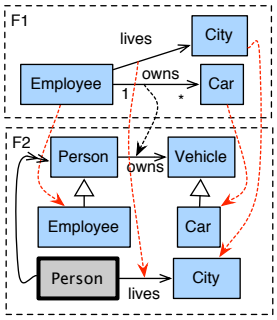
A fragment (see Fig. 4a) is a pair $F = (SG, tr)$, comprising an *SG* plus a target function for reference edges (def. 9 introduces set *Fr*, $F \in Fr$). Referencing (through *tr*) is illustrated in Fig. 4: in fragment `F2` of Fig. 4b, proxy `Person` (thick

[2]In Isabelle, we proved that src^* and tgt^* preserve the information of base source and target functions; see [4].

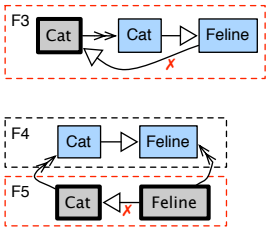
[3]This is because SGs require that all nodes pertain to the graph, not allowing references that may be located in other graphs



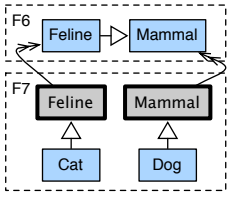
(a) Local Fragments



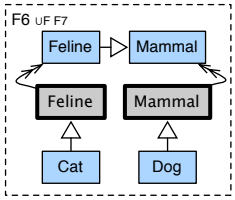
(b) A Fragment morphism



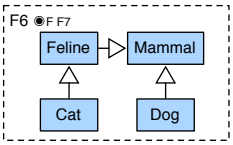
(c) Fragments with inheritance cycles



(d) Two valid fragments

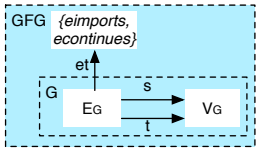


(e) Union fragment composition

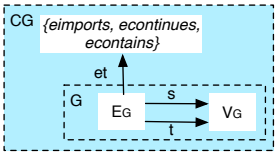


(f) Colimit fragment composition

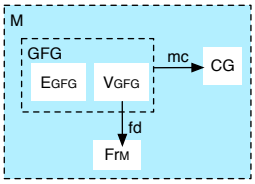
Fig. 4: Fragments



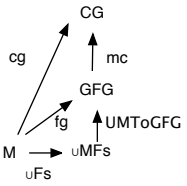
(a) Global Fragment Graphs



(b) Cluster Graphs



(c) Structure of models



(d) Tower of morphisms of a model

Fig. 5: Global fragment graphs, cluster graphs and models

line) refers to node with same name, likewise for Figs 4c, 4d and 4e. A referred node may be either in the proxy's fragment or elsewhere (`F2` contains an intra-fragment reference, and `F5` and `F7` contain inter-fragment references). Three different representations can be extracted: (i) an unreferenced graph (SG view), (ii) a graph with referenced proxies only, and (iii) a referenced graph.

FRAGMENTA forbids inheritance cycles, such as the ones illustrated in Fig. 4c: `F3` contains an explicit (direct) cycle that is excluded through a constraint deeming the inheritance relation enriched with references to be acyclic, and `F4` together with `F5` contain a semantic (indirect) cycle that is excluded by stating that proxy nodes cannot have supertypes – see def. 9 for details. In Isabelle, we proved that the local fragment constraints preclude both local and global inheritance cycles (see fact 1 in appendix).

FRAGMENTA provides a composition based on set-union to put fragments together without resolving the references (def. 10), as illustrated in Fig. 4e with fragments `F6` and `F7` of Fig. 4d. The composition that resolves the references (called *colimit composition*, section V below) is illustrated in Fig. 4f. The inheritance edges of proxies in Figs. 4d and 4e are valid: proxies may not have supertypes, but subtypes are allowed.

Fragment morphisms (F-morphisms) handle the semantics of reference edges, which is akin to inheritance: an edge attached to a node is an edge of that node and all its representations in the fragment. In `F2` of Fig. 4b, edges `lives` and `owns` pertain to both nodes named `Person`. To support this, fragments extend relations $\prec$, $\preccurlyeq$, src^* and tgt^* of SGs to cover the semantics of references[4]. This extension is based on functions *refs*, which gives the references relation between proxies and their referred nodes (obtained from a restricted graph with reference edges only), and function $\rightsquigarrow$, which yields a relation giving all the representatives of a given node ($\rightsquigarrow_F = refs_F \cup (refs_F)^\sim$), and the actual inheritance relation for fragments, which extends the inheritance of SGs with the representatives relation ($\prec_F = \prec_{sg\,F} \cup \rightsquigarrow_F$) [4].

F-morphisms (def. 11) are similar to SG-morphisms, but taking references into account using the extended relations. In Isabelle, we proved the correctness of the F-morphism of Fig. 4b and the one in the inverse direction.

B. Global Fragment and Cluster Graphs

Global fragment graphs (GFGs) represent fragment relations. A GFG (Fig. 5a) is a pair $GFG = (G, et)$ made of a graph and an edge colouring function, stating whether the edge is an imports or continues (set *GFGr* of def. 12, $GFG \in GFGr$). Graph `GFG_MONDO_M` of Fig. 6 is a GFG-specimen. We define two sets of morphisms for GFGs: GFG-morphisms (def. 13), which preserve edge-colouring, and fragment to GFG morphisms [4] from fragment local nodes to their corresponding global fragment nodes.

A *cluster graph* (CG) identifies clusters and their relations. As shown in Fig. 5b, a CG is a pair $CG = (G, et)$ made of a graph G and an edge colouring function et, stating whether the related clusters are in a relation of imports, continues or contains (def. 14 introduces set CGr, $CG \in CGr$); the

[4]In Isabelle, we proved that the extensions preserve the information of the corresponding SG relation (e.g $\preccurlyeq_F \subseteq \preccurlyeq_{sg\,F}$); see [4].

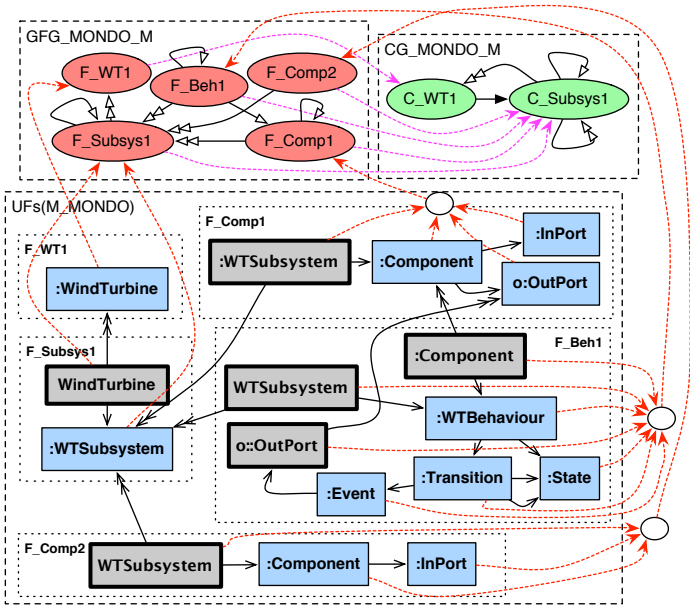
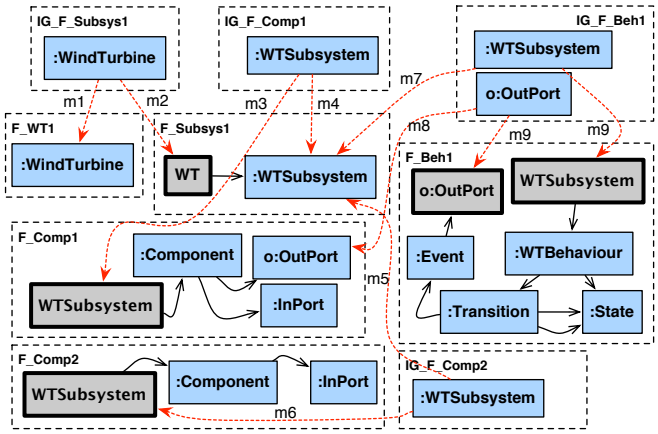
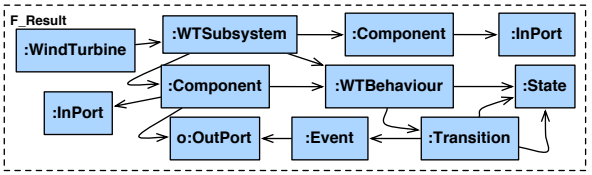


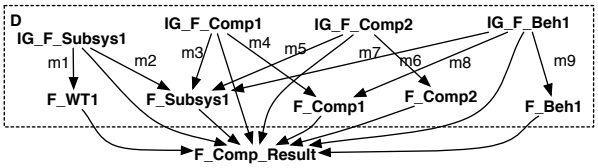
Fig. 6: Model highlighting underlying morphisms. The bottom graph describes the union of all fragments of the model



(a) Machinery of colimit composition



(b) Result of colimit composition



(c) Colimit composition diagram

Fig. 7: FRAGMENTA's colimit composition

relation formed by the contains edges must form a forest. Graph `CG_MONDO_M` of Fig. 6 is a CG. We define two sets of morphisms for CGs: CG-morphisms [4] and GFG to CG morphisms [4]; both preserve edge-colouring.

C. Models

A FRAGMENTA model is a collection of fragments. As shown in Fig. 5c, a model is a tuple $M = (GFG, CG, mc, fd)$, comprising a *GFG*, a *CG*, a morphism mc: $GFG \rightarrow CG$, and a function $fd : Ns_{GFG} \rightarrow Fr$ mapping nodes of the GFG to fragment definitions (Fr is set of all fragments) – def. 15 introduces set Mdl, $M \in Mdl$. In Fig. 5c, Fr_M is the set of fragments of a model, as given by the range of fd. Each fragment has its own nodes and edges.

As outlined in Fig. 3, FRAGMENTA's models have three layers. Hence, each model has an underlying tower of morphisms relating these three layers. Fig. 5d depicts this: from a model M, we can obtain the union of all the model's fragments (function *UFs*), and from this we can construct a morphism to the model's GFG (function *UMToGFG*), and from here the model's morphism mc gets to the model's *CG*. Figure 6 illustrates this: `M_MONDO` at the bottom is the fragment resulting from *UFs* (union of all model fragments).

V. MODEL COMPOSITION AS REFERENCE RESOLUTION

The previous section highlighted FRAGMENTA's overall model built as the union of all fragments (fragment `M_MONDO` in Fig. 6). This constitutes a simple form of composition: the overall model retains proxy nodes and their references.

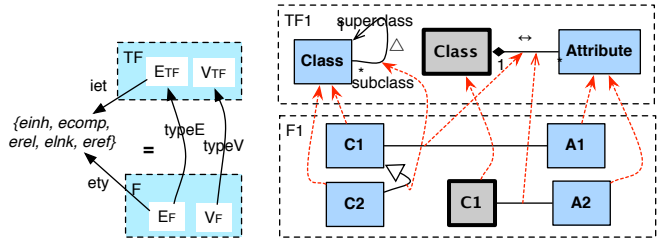
This section presents fragment composition as a process of reference resolution: proxy and referred nodes are merged, and reference edges eliminated. This is based on the *colimit* construction of category theory. All details of colimit composition are given in [4]. Here, we outline the approach using the example of Fig. 6 (whose composition is given in Fig. 7b):

- We construct *interface graphs* (IGs) for each fragment, only containing proxies. This is illustrated in Fig. 7a (graphs named `IG_F...`).
- For each IG, we construct morphisms from the reference edges, using the source and target reference functions of the fragment. In Fig. 7a, we have morphisms that map node `:WT` of `IG_F_Subsys1` to nodes with same name in `F_WT1` and `F_Subsys1` (the target reference and source of corresponding reference edge, respectively).
- Following this scheme, we build a diagram of IGs and SGs without reference edges corresponding to the fragments being composed as show in Fig. 7c.
- By applying the colimit to all the graphs of a diagram, we obtain a SG without references as shown in Fig. 7b.

VI. TYPING AND FRAGMENTATION STRATEGIES

A. Typed Fragments

The core of FRAGMENTA's typing approach lies at the fragment level. This covers both the local and global realms; as shown in sec. IV-A, global properties (including conformance) are then considered in the realm of a global fragment that is built as the union of all fragments of a model.



(a) Typed fragment structure (b) Typed fragment example

Fig. 8: Typed fragment structure (a) and example (b). In (b) edges of type fragment are decorated with prescribed edge type ($\bigtriangleup$: inheritance; $\leftrightarrow$: relation)

We introduce two structures to represent fragment typing:

- A type fragment, $TF = (F, iet)$, comprises a fragment F and a colouring funcion $iet : EsA_F \rightarrow SGET$, giving the type of instance edge being prescribed (def. 16).
- A typed fragment (Fig. 8a), $FT = (F, TF, type)$, is made of an instance F, a type fragment TF and a morphism $type : Fr \rightarrow TFr$, mapping instance to type (def. 17).

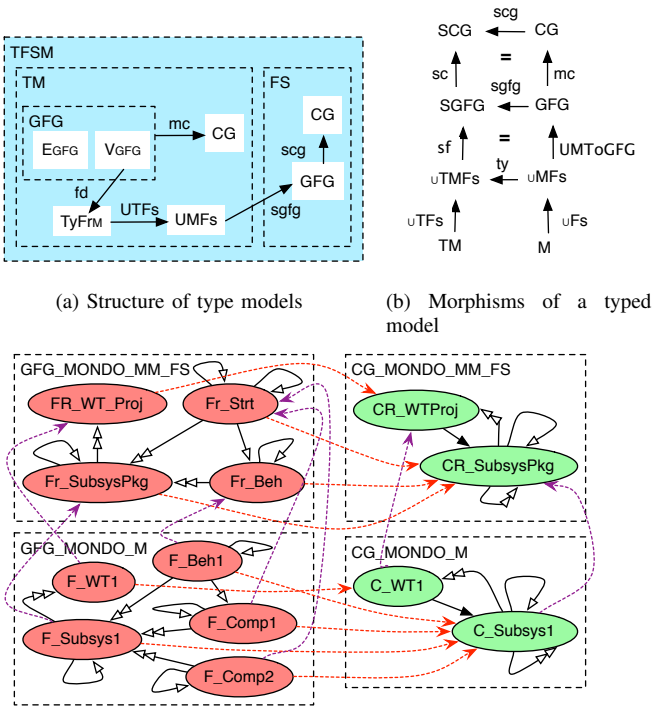
Fig. 8b presents a typed fragment, describing a simple class model that includes proxies.

Section IV-A introduced a relaxed notion of fragment morphism. This covers a variety of model relations at same and different meta-levels, but misses certain typing specificities, such as multiplicities. To complement F-morphisms, we introduce the notion of type conformance to check that the instance conforms to the constraints imposed by the type. The conformance constraints are: (a) edge types of instance fragment conform to those prescribed by type fragment (commutativity of diagram in Fig. 8a); (b) abstract nodes may not have direct instances; (c) containments are not shared; (d) multiplicity constraints; and (e) the relation formed by instances of containment edges forms a forest. These constraints cater to proxy nodes (as illustrated in Fig. 8b).

B. Typed Models with Fragmentation Strategies

Model typing builds up on fragment typing and FSs enrich model typing. The following support model typing and FSs:

- A FS is a tuple $FS = (GFG_S, CG_S, sc, sf)$, comprising the FS's CG (cluster regions), a FS's GFG (fragment regions), and morphisms sc (GFG_S to CG_S) and sf (model fragment elements to GFG_S) – illustrated in Fig. 1a.
- A type model (a fragmented metamodel) differs from a model (section IV-C) in that it uses type rather than plain fragments. A type model with FS, depicted in Fig. 9a, is a tuple $TFSM = (TM, FS)$, containing a type model $TM = (GFG, CG, mc, fd)$ and a FS (see [4]).
- A typed model puts together type and instance models. It is a tuple $MT = (M, TM, scg, sgfg, ty)$, made of a model M, a type model TM and three morphisms: (i) scg maps CG of M into the FS's CG of TM, (ii) $sgfg$ maps GFG of M into the FS's GFG of TM, and (iii) ty maps model elements of M into their types in TM. Typed models and their morphisms are depicted in Fig. 9b.



(a) Structure of type models (b) Morphisms of a typed model

(c) CG and GFG Morphisms from model into metamodel's FS

Fig. 9: Typed FRAGMENTA models

A typed model requires the commutativity of diagrams in Fig. 9b, entailing FS conformance (morphisms scg and $sgfg$) and typing (ty, through union of fragments of M and TM).

Fig. 9c depicts the morphisms that exist between a model's CG and GFG and their counterparts in the metamodel's FS for the example of Fig. 1. The top graphs describe the cluster and fragment regions of the FS of Fig.1a.

VII. DISCUSSION

Modular design. FRAGMENTA aims to support separation of concerns effectively. This, however, brings a complexity cost to the underlying theory. SGs, with their support for inheritance, add complexity to plain graphs; fragments, with their proxies, add further complexity to SGs. FRAGMENTA hides this complexity to enable design of fragmented models that harness separation of concerns. The support for both top-down and bottom-up design means that designers can choose the scheme that best suits their problems and way of thinking. This is realised through FRAGMENTA's concepts of continuations and imports that are variations on how proxies and their references are interpreted at upper level of GFGs.

To gain the important result of global preservation of inheritance acyclicity checked locally (fact 1), we forbid proxies with supertypes. We do not see this as a serious restriction. It can be seen as a design rule whereby a concept's supertypes must be defined when the concept is first introduced; proxies

may then have subtypes, but no supertypes. In the end, what we gain is greater than what we loose, given the applicability of the result at both meta and instance levels, and the prevalent use of inheritance in MDE- and DSL-based modelling.

A theory of separation. Section V presented colimit-based model composition, which resolves references through substitution. FRAGMENTA, however, keeps the models fragmented. The compositions that are required for global purposes are based on the union of all model fragments without reference resolution, a simpler operation. FRAGMENTA lives well with separation; its machinery handles a world where a concept may be represented by many nodes, in contrast with monolithic approaches that support one node per concept only. We envision the resolution compositions outlined in sec. V as being an aid to designers to get a clean big picture.

The definition of fragments connects proxies to their referring nodes through a function (*tr*, def. 9), which does not preclude or impede use of fragments in isolation. This function may be implemented externally to the fragment definition.

Fragmentation strategies complement metalevel definitions of types with definitions of fragmentation structure. This ensures uniform fragmentations across model instances, which is useful when dealing with big models and collections of related models. This paper's running example (Fig. 1) illustrates usefulness of FSs concept; the different models of wind-turbine controllers should have a uniform structure. Often, such uniformities are agreed among developers with no means to express or enforce them, which complicates the processing of models, introducing accidental complexity. Our approach formally defines FSs so that their conformity can be enforced and checked by tools. In our theory, such conformances are described as a commuting of instance and type diagrams, as shown in Fig. 9b.

FRAGMENTA's realisations. FRAGMENTA and its underlying ideas have been implemented in two Eclipse-based tools as part of EU project MONDO: (i) *DSL-tao* [5] enables the pattern-based construction of DSL meta-models and their supporting modelling environments, supporting FRAGMENTA's concepts of fragment and cluster; (ii) *EMF-Splitter* [6] implements the notion of FS proposed here[5]. FRAGMENTA can also be used as a modularity paradigm with the notions of cluster and fragments realised in its many guises. VCL's [7], [8] modularity mechanisms resemble FRAGMENTA. In VCL, FRAGMENTA's clusters are packages and fragments are VCL diagrams. VCL does not provide any support for top-down design. FRAGMENTA's contructions could greatly simplify the design of a modelling language such as VCL.

Machine-assisted specification and proof. FRAGMENTA was specified in the Z language and its consistency was checked using the CZT typechecker to ensure consistency with respect to names and types. Z's expressivity, grounded on its mathematical generality, high-order capabilities and its Zermelo-Fraenkel set-theory underpinning (a widely accepted foundation of mathematics), enabled us to describe FRAGMENTA's mathematical definitions based on graphs, functions, sets, relations and categories. This powerful expressivity was known to us based on our prior experience with Z. The Z specification (very close to the presentation given here and provided in [4]) was then encoded in the state of the art Isabelle proof assistant[6] and its expressive high-order logic. This step required some meaning-preserving changes to cater to Isabelle's specificities (e.g., Isabelle's lack of partial function primitive). Isabelle was used to validate and verify FRAGMENTA; we proved general theorems concerning desired properties (verification) and theorems concerning examples (validation). Table I gives the number of Isabelle proofs that were undertaken.

Verification	268
Validation	123
Total	**391**

TABLE I: Number of Isabelle proofs undertaken to validate and verify FRAGMENTA.

The real world. Our case studies [4] include the industrial language used here and several examples drawn from VCL [7], [8], a medium sized modelling language. FRAGMENTA's SGs are an abstraction of MDE structural models, supporting inheritance, composition and multiplicities. FRAGMENTA's proxies are an abstraction of EMF proxies [2] and VCL's referencing mechanism. Our proved result (fact 1) showing that the well-formedness of a inheritance hierarchy (acyclicity) checked locally at the fragment level is preserved globally (provided some local constraints are met, namely that proxies may not have supertypes) is relevant for the current practice due to the popularity of EMF; this means that any code that is generated from a FRAGMENTA-compliant structure of models and metamodels is guaranteed to be free of compilation errors concerning inheritance well-formedness.

FRAGMENTA's three-level architecture can capture the tree-based structure of modern modelling and programming projects; in terms of a file system, fragments can be mapped to files and clusters to folders.

Formalisation. FRAGMENTA formalises inheritance using coloured edges in SGs, as any other edge, unlike similar graphs [9], [10], which capture inheritance as a relation. The edge solution gives uniformity to our theory and makes inheritance amenable to typing (as illustrated in Fig. 8b); our edge-colouring solution also simplifies checking the prescribed edge type to a simple diagram commuting (Fig. 8a).

A formalisation of references as coloured edges was chosen in detriment of a partial function ($refs : V \nrightarrow V$). This choice benefits FRAGMENTA's uniformity, coherence (all edges are formalised as such) and clarity (such edges appear in the morphisms from local fragment nodes to GFGs as inter-fragment GFG edges). The drawback of reference edges is that they lie unreferenced in SGs, requiring use of the reference target function of fragments to get graphs that are referenced.

VIII. RELATED WORK

There is widespread acknowledgement of MDE's scalability challenge and the need for modularity. The popular EMF

[5]DSL-Tao: http://bit.ly/1CPTYZd. EMF-Splitter: http://bit.ly/1Eq1TZD

[6]The Isabelle theories can be found at http://www.miso.es/fragmenta/

provides the means to partition models with proxies, but lacks support for fragmentation strategies (FSs). To improve this, [11] proposes a non-formal persistence framework for EMF to fragment models along annotated metamodel compositions. Our theory is formal and provides a powerful notion of fragmentation regions that allows metamodel-defined fragmentations along our container primitive of clusters.

Heidenreich et al [12] propose a non-formal language independent modularisation approach that puts together fragments through composition interfaces made of reference and variation points. FRAGMENTA is more abstract than [12]; it provides a mathematical notion of joints based on proxys and their references, similar to the reference points of [12], that is amenable to model composition based on the general colimit.

Weisemöller and Schürn [13] try to improve the modularisation of MOF, a popular metamodelling language. Their formalisation introduces metamodel components equipped with export and import interfaces to enable composition. Their definition of metamodel equates to the simple graphs presented here, not considering important concepts such as inheritance, composition and multiplicities. Furthermore, [13] deals with metamodels only; FRAGMENTA covers both levels, not making a substantial distinction between models and metamodels.

Certain formal approaches to *merge composition* [14], [15] also use the colimit construction of category theory. Our work does a more thorough treatment of the proxy mechanism for referencing and incremental definition, which is slightly different from the merge, and puts forward the simpler union composition, where references are not resolved.

Hermann et al [9] investigate inheritance in a graph transformation setting, considering a special condition in metamodel morphisms to ensure existence of co-limits of arbitrary categorical diagrams. FRAGMENTA does not perform co-limits over arbitrary diagrams, considering only those that are related through proxies (interface graphs, see Fig. 7). Although related, settings of [9] and FRAGMENTA are different; [9] is not concerned at all by inheritance acyclicity and proxies.

Component graphs [10] with its two-layer structuring, local and network, resemble FRAGMENTA's local and global fragment levels. FRAGMENTA provides an extra third level of clusters. [10] provides IC-graphs, which are similar to SGs but without multiplicities, and uses import and export interfaces to enable composition. FRAGMENTA uses proxies to build fragments incrementally in either a bottom-up or top-down fashion, which is closer to EMF proxies. [10] acknowledges how such graph structures are capable of capturing the EMF, but without providing a formal study of proxies (an EMF concept). [10] also acknowledges that inheritance well-formedness issues (cycles) may arise when parts are composed, but there is no proved result, like the one presented here, concerning the global preservation of inheritance well-formedness (acyclicity, fact 1) provided some local constraints are met.

Hamiaz et al [16] formalise in the Coq theorem prover the model composition operations of [12]. This shares FRAGMENTA's emphasis on formalisations developed with proof assistants. FRAGMENTA, however, is more abstract; it is a general approach that mimics common features of MDE; composition is expressed in terms of general mathematical operators, such as colimit and set-union.

Several approaches split monolithic models. Kelsen et al [17] propose an algorithm to split a model into submodels, where each submodel is conformant to the original metamodel with association multiplicities taken into account. Strüber et al [18] provide a splitting mechanism for both metamodels and models based on the component graphs of [10]. In [19], Strüber et al use [10] as the basis of an approach to split a model based on the relevance of its elements using information retrieval methods. Unlike these works, FRAGMENTA is a design theory, supporting the novel idea of metamodel defined FSs and a hierarchical organisation of fragments into clusters.

IX. Conclusions and future work

This paper presented FRAGMENTA, a formal theory to fragment MDE models. This paper's main result (fact 1), formally derived from the theory, is that the satisfaction of some local fragments constraints (particularly, the fact that proxies may not have supertypes) is enough to ensure that inheritance hierarchies remain well-formed (acyclic) globally when fragments are composed. This means that implementations complying with FRAGMENTA's constraints will be free of inheritance cycle errors. This is relevant because the widely diffused EMF uses a similar proxy mechanism and inheritance is prevalent in current practice.

FRAGMENTA's main novelties include: (a) the formal treatment of model fragments exploiting the particularities of a seaming mechanism based on proxies, (b) metalevel fragmentation strategies that stipulate a fragmentation structure to model instances, (c) support for both bottom-up and top-down fragmented designs and (d) three-level model architecture. Other minor novelties include: (i) the observation that although fragmented models are amenable to colimit-based composition, this operation is not necessary for the theory's internal global processing, which can live with unresolved references; and (ii) fragment graphs and the way they capture the proxy concept.

FRAGMENTA was developed with the assistance of tools, using specification type-checkers and proof assistants. Two tools, DSL-tao and EMF-Splitter, were developed based on FRAGMENTA. We are currently working on FRAGMENTA's merging mechanisms, further developing the tools and applying the theory to additional case studies.

Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competivity (TIN2011-24139 and TIN2014-52129-R), Madrid's Region R & D programme (S2013/ICE-3006), and the EU project MONDO (FP7-ICT-2013-10, #611125).

References

[1] P. Tarr, H. Ossher, W. Harrison, and J. Stanley M. Sutton, "N Degrees of Separation: Multi-Dimensional Separation of Concerns," in *ICSE'99*.

[2] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*. Addison-Wesley, 2008.

[3] T. Nipkow, L. C. Paulson, and M. Wenzel, *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2002.

[4] N. Amálio, J. de Lara, and E. Guerra, "Fragmenta: a theory of separation to design fragmented MDE models," UAM, Tech. Rep., 2015. [Online]. Available: http://bit.ly/1rHlxrc

[5] A. Pescador, A. Garmendia, E. Guerra, J. S. Cuadrado, and J. de Lara, "Pattern-based development of domain-specific modelling languages," in *MODELS 2015*, (this volume).

[6] A. Garmendia, E. Guerra, D. S. Kolovos, and J. de Lara, "EMF splitter: A structured approach to EMF modularity," in *Proc. XM@MODELS*, ser. CEUR Workshop Proceedings, vol. 1239, 2014, pp. 22–31.

[7] N. Amálio, P. Kelsen, Q. Ma, and C. Glodt, "Using VCL as an aspect-oriented approach to requirements modelling," *TAOSD*, vol. 7, pp. 151–199, 2010.

[8] N. Amálio and C. Glodt, "A tool for visual and formal modelling of software designs," *Science of Computer Programming*, vol. 98, Part 1, pp. 52 – 79, 2015.

[9] F. Hermann, H. Ehrig, and C. Ermel, "Transformation of type graphs with inheritance for ensuring security in e-government networks," in *FASE 2009*.

[10] S. Jurack and G. Taentzer, "Transformation of typed composite graphs with inheritance and containment structures," *Fundam. Inform.*, vol. 118, no. 1-2, pp. 97–134, 2012.

[11] M. Scheidgen, A. Zubow, J. Fischer, and T. H. Kolbe, "Automated and transparent model fragmentation for persisting large models," in *MODELS 2012*, ser. LNCS, vol. 7590. Springer, 2012, pp. 102–118.

[12] F. Heidenreich, J. Henriksson, J. Johannes, and S. Zschaler, "On language-independent model modularisation," *TAOSD VI*, vol. 6, pp. 39–82, 2009.

[13] I. Weisemöller and A. Schürr, "Formal definition of MOF 2.0 metamodel components and composition," in *MODELS 2008*.

[14] S. Nejati, M. Sabetzadeh, M. Chechik, S. Easterbrook, and P. Zave, "Matching and merging of statecharts specifications," in *ICSE'2007*.

[15] M. Sabetzadeh and S. Easterbrook, "An algebraic framework for merging incomplete and inconsistent views," in *RE 2005*. IEEE, 2005.

[16] M. K. Hamiaz, M. Pantel, B. Combemale, and X. Thirioux, "Correct-by-construction model composition: Application to the invasive software composition method," in *Proc. FESCA@ETAPS*, ser. EPTCS, 2014.

[17] P. Kelsen, Q. Ma, and C. Glodt, "Models within models: Taming model complexity using the sub-model lattice," in *FASE 2011*.

[18] D. Strüber, G. Taentzer, S. Jurack, and T. Schäfer, "Towards a distributed modeling process based on composite models," in *FASE 2013*.

[19] D. Strüber, J. Rubin, G. Taentzer, and M. Chechik, "Splitting models using information retrieval and model crawling techniques," in *FASE 2014*.

APPENDIX

This appendix gives an abridged presentation of [4]. Each structure that is introduced has an associated set of functions, predicates and proof laws that are given in full in [4].

Definition 1. *The disjoint sets V and E represent all possible nodes and all possible edges of graphs, respectively.* □

Definition 2. *A graph $G = (V_G, E_G, s, t)$ (Fig. 2a) consists of sets $V_G \subseteq V$ of nodes and $E_G \subseteq E$ of edges, and source and target functions $s, t : E_G \to V_G$.*

The set of graphs Gr, such that $G : Gr$, is defined as:

$$Gr = \{(V_G, E_G, s, t) \mid V_G \in \mathbb{P}\, V \land E_G \in \mathbb{P}\, E \land s \in E_G \to V_G \land t \in E_G \to V_G\}$$

In the following, we write Ns_G, Es_G, src_G and tgt_G to refer to the nodes, edges, source and target functions of a graph G, respectively (defined as functions in [4]). Predicate $disj(G_1, G_2)$ says whether two graphs are disjoint (no nodes or edges in common, defined in [4]). □

Definition 3. *A graph morphism $m : G_1 \to G_2$ defines a mapping between graphs $G_1, G_2 : Gr$; it comprises a pair of functions $m = (f_V, f_E)$, $f_V : Ns_{G1} \to Ns_{G2}$ and $f_E : Es_{G1} \to Es_{G2}$, mapping nodes and edges respectively that preserve the source and target functions of edges: $fv \circ src_{G_1} = src_{G_2} \circ fe$ and $fv \circ tgt_{G_1} = tgt_{G_2} \circ fe$ (see Fig. 2b). Sets GrMorph (all possible graph morphisms) and $G_1 \to G_2$ (morphisms between two graphs), such that $G_1 \to G_2 \subseteq GrMorph$, are defined as:*

$$\begin{aligned} &GrMorph = \{(fv, fe) \mid fv \in V \nrightarrow V \land fe \in E \nrightarrow E\} \\ &G_1 \to G_2 = \{(fv, fe) \mid fv \in Ns_{G_1} \to Ns_{G_2} \land fe \in Es_{G_1} \to Es_{G_2} \\ &\quad \land fv \circ src_{G_1} = src_{G_2} \circ fe \land fv \circ tgt_{G_1} = tgt_{G_2} \circ fe\} \end{aligned}$$

Above, the two equations involving function composition (symbol $\circ$) ensure diagram commutativity (depicted in Fig. 2b). □

Definition 4. *The composition of graph morphisms $f : G_1 \to G_2$ and $g : G_2 \to G_3$, $G_{i \in \{1,3\}} : Gr$, is defined as:*

$$g \circ_G f = ((fv\, g) \circ (fv\, f), (fe\, g) \circ (fe\, f))$$

□

Definition 5. *The node types of a SG (set SGNT) are: normal, abstract and proxy. The edge types (set SGET) are: inheritance, containment, relation, link and reference:*

$$SGNT = \{nnrml, nabst, nprxy\} \qquad SGET = \{einh, ecomp, erel, elnk, eref\}$$

□

Definition 6. *Sets MultUVal (upper bound values) and Mult (multiplicities) are defined below. MultUVal is disjoint union (symbol $\uplus$) of natural numbers and singleton set with $*$ (*many*); Mult is a set of lower and upper bound pairs.*

$$\begin{aligned} &MultUVal = \mathbb{N} \uplus \{*\} \\ &Mult = \{(lb, ub) \mid lb \in \mathbb{N} \land ub \in MultUVal \land (ub = * \lor (ub \in \mathbb{N} \land lb \leq ub))\} \end{aligned}$$

□

Definition 7. *A structural graph $SG = (G, nty, ety, sm, tm)$ comprises a graph $G : Gr$, two colouring functions for nodes and edges, $nt : Ns_G \to SGNT$ and $et : Es_G \to SGET$, and source and target multiplicity functions, $sm, tm : Es_G \nrightarrow Mult$ (Fig. 2c).*

Base set SGr_0 of SGs, such that $SG : SGr_0$, is defined as:

$$\begin{aligned} SGr_0 = \{&(G, nt, et, sm, tm) \mid G \in Gr \land nt \in Ns_G \to SGNT \\ &\land et \in Es_G \to SGET \land sm \in Es_G \nrightarrow Mult \land tm \in Es_G \nrightarrow Mult\} \end{aligned}$$

Actual set of SGs, SGr, is defined from the base set, using functions and predicates of [4], as:

$$\begin{aligned} SGr = \{&SG : SGr_0 \mid EsR_{SG} \subseteq EsId_{SG} \\ &\land srcm_{SG} \in EsTy(SG, \{erel, ecomp\}) \to Mult \\ &\land tgtm_{SG} \in EsTy(SG, \{erel, ecomp\}) \to Mult \\ &\land srcm_{SG}\, (\!|\, EsTy(SG, \{ecomp\})\, |\!) = \{(0,1), 1\} \land acyclicI\, SG\} \end{aligned}$$

SGs have the following constraints: (a) reference edges (EsR_{SG}) are self edges ($EsId_{SG}$), (b) relation and containment edges must have multiplicities, (c) source multiplicity of containment edges should be $0..1$ or 1, and (d) the inheritance graph must be acyclic (predicate acyclicI). □

Definition 8. *Given SGs $SG_1, SG_2 : SGr$, a SG morphism $m : SG_1 \to SG_2$ is a pair of functions $m = (fv, fe)$ mapping nodes and edges, respectively. The set of morphisms between two SGs, $SG_1 \to SG_2$, is defined as:*

$$\begin{aligned} SG_1 \to SG_2 = \{&(fv, fe) \mid fv \in Ns_{SG_1} \to Ns_{SG_2} \land fe \in Es_{SG_1} \to Es_{SG_2} \\ &\land fv \circ src^*_{SG_1} \subseteq src^*_{SG_2} \circ fe \land fv \circ tgt^*_{SG_1} \subseteq tgt^*_{SG_2} \circ fe \land fv \circ \preccurlyeq_{SG_1} \subseteq \preccurlyeq_{SG_2} \circ fv\} \end{aligned}$$

This states subset commuting ($\subseteq$ rather than $=$) of underlying graphs to preserve inheritance constraints (depicted in Figs. 2d and 2e); here, $\circ$ is relation composition. □

Definition 9. *A fragment $F = (SG, tr)$ (depicted in Fig. 4a) comprises a $SG : SGr$, and a function $tr : EsRP_{SG} \to V$, mapping reference edges attached to proxies to referred nodes.*

The base set of local fragments Fr_0 is defined as:

$$Fr_0 = \{(SG, tr) \mid SG \in SGr \wedge tr \in EsRP_{SG} \to V \\ \wedge (EsRP_{SG} \lhd src_{SG}) \in EsRP_{SG} \rightarrowtail NsP_{SG} \\ \wedge EsTy(SG, \{einh\}) \lhd src_{SG} \rhd NsP_{SG} = \varnothing\}$$

Above, $\lhd$ and $\rhd$ are domain and range restrictions, respectively. The constraints say that (i) tr is a total function from reference edges attached to proxies (EsRP) to some node, (ii) that the references edges are attached to at most one proxy, (iii) and that proxy nodes (NsP_{SG}) cannot have supertypes.

Set of fragments Fr extends base set using functions of [4]:

$$Fr = \{F \mid F \in Fr_0 \wedge (\forall\, v : NsP_F \bullet nonPRefsOf\ F\ v \neq \varnothing) \wedge acyclicIF\ F\}$$

This states that, ultimately, proxies must reference non-proxy nodes and that the extended inheritance relation is acyclic. □

Definition 10. *The union composition of fragments $F_1, F_2 : Fr$ is the union of the fragments' SGs (function sg and operator $\cup_{SG}$ of [4]) and union of target reference functions (tgtr):*

$$F_1 \cup_F F_2 = (sg\ F_1 \cup_{SG} sg\ F_2, tgtr_{F_1} \cup tgtr_{F_2})$$ □

Fact 1. *Given fragments $F1, F2 : Fr$, we have the following:*

- *The union of two disjoint fragments is inheritance acyclic provided that individually the fragments are acyclic also:*

$$F_1 \in Fr;\ F_2 \in Fr;\ disjFs(F_1, F_2) \\ \vdash acyclicIF\ (F_1 \cup_F F2) \Leftrightarrow acyclicIF\ F_1 \wedge acyclicIF\ F2$$

- *The union of two disjoint fragments is well-formed provided the individual fragments are well-formed also:*

$$disjFs(F_1, F_2) \vdash (F_1 \cup_F F2) \in Fr \Leftrightarrow F_1 \in Fr \wedge F2 \in Fr$$

- *Every fragment obtained after resolving the references is well-formed (and hence acyclic).*

Proof. *These three theorems were proved in Isabelle[7]; proof outlines can be obtained from [4].* □

Definition 11. *Given fragments $F_1, F_2 : Fr$, a fragment morphism $m : F_1 \to F_2$ is a pair of functions $m = (fv, fe)$ mapping nodes and edges, respectively. The set of such morphisms is:*

$$F_1 \to F_2 = \{(fv, fe) \mid fv \in Ns_{F_1} \to Ns_{F_2} \wedge fe \in Es_{F_1} \to Es_{F_2} \\ \wedge fv \circ src^*_{F_1} \subseteq src^*_{F_2} \circ fe \wedge fv \circ tgt^*_{F_1} \subseteq tgt^*_{F_2} \circ fe \wedge fv \circ \preccurlyeq_{F_1} \subseteq \preccurlyeq_{F_2} \circ fv\}$$

Above, we restate the same conditions as SG morphisms (def. 8), using the updated functions and relations for fragments (see [4]) that cater to the semantics of references. □

Definition 12. *The set of extension edge kinds (imports and continues) is:*

$$ExtEdgeTy = \{eimpo, econti\}$$

[7]The first theorem shows impossibility of direct cycles (as per F3, Fig. 4c) in union fragment compositions, the second theorem is a closure property of fragment union, and third theorem shows impossibility of indirect cycles (as per F4 and F5 in Fig. 4c) in a well-formed fragment

A global fragment graph (GFG) is a pair $GFG = (G, et)$, where $G : Gr$ is a graph (def. 2), and $et : Es_G \to ExtEdgeTy$ is a colouring function mapping edges to extension edge types. The imports and continues relations taken together (edges of graph) and excluding self edges must be acyclic. Set of valid GFGs, GFGr, is defined as:

$$GFGr = \{(G, et) \mid G \in Gr \wedge et \in Es_G \to ExtEdgeTy \\ \wedge acyclicG(restrict(Es_G \setminus EsId_G))\}$$ □

Definition 13. *Given $GFG_1, GFG_2 : GFGr$ (def. 12), a GFG morphism $m : GFG_1 \to GFG_2$ defines a mapping between them. The set of GFG-morphisms is defined as:*

$$GFG_1 \to GFG_2 = \{m \mid m \in gr\ GFG_1 \to gr\ GFG_2 \wedge ety_{GFG_2} \circ fE\ m = ety_{GFG_1}\}$$

This requires that GFG morphisms are normal graph morphisms (function gr) that preserve colouring of edges. □

Definition 14. *Set of cluster edge kinds is extension edge kinds (ExtEdgeTy, def. 12) plus containment:*

$$CGEdgeTy = ExtEdgeTy \cup \{econta\}$$

A cluster graph (CG) is a pair $CG = (G, et)$, comprising a graph $G : Gr$ (definition 2) and a colouring function $et : Es_G \to CGEdgeTy$ mapping edges to cluster edge types. The set of valid cluster graphs CGr is defined as:

$$CGr = \{(G, et) \mid G \in Gr \wedge et \in Es_G \to CGEdgeTy \\ \wedge acyclicG(restrict(G, et^{\sim}(\!|\{eimpo, econti\}|\!) \setminus EsId_G)) \\ \wedge rel(restrict(G, et^{\sim}(\!|\{econta\}|\!) \setminus EsId_G)) \in forest\}$$

This states: (i) relation formed by the imports and continues edges (together), subtracted with self edges, must be acyclic, and (ii) relation formed by containment edges, subtracted with self edges, must constitute a forest. □

Definition 15. *A model is quadruple $M = (GFG, CG, mc, fd)$, consisting of a $GFG : GFGr$, a $CG : CGr$, a morphism $mc : GFG \to CG$, and a mapping from GFG nodes to fragment definitions $fd : Ns_{GFG} \to Fr$. Base set of models Mdl_0 is:*

$$Mdl_0 = \{(GFG, CG, mc, fd) \mid GFG \in GFGr \wedge CG \in CGr \\ \wedge mc \in GFG \to CG \wedge fd \in Ns_{GFG} \to Fr\}$$

Set of all models extends base set using definitions of [4]:

$$Mdl = \{M : Mdl_0 \mid UMToGFG\ M \in UFs\ M \to (gfg\ M) \\ \wedge (\forall\, vf_1, vf_2 : Ns(gfg\ M) \mid vf1 \neq vf2 \bullet \text{disjFs}(fdef_M\ vf_1, fdef_M\ vf_2))\}$$

This says that morphism obtained from UMToGFG maps a fragment (union of model's fragments) to a GFG, and that all fragments of model are disjoint (predicate disjFs*).* □

Definition 16. *A type fragment is a pair $TF = (F, iet)$ that comprises a fragment $F : Fr$ and a colouring function $iet : EsA_F \to SGET$ that indicates stipulated instance edge type. Set TFr, such that $TF : TFr$, is defined as:*

$$TFr = \{(F, iet) \mid F \in Fr \wedge iet \in EsA_F \to SGET\}$$ □

Definition 17. *A typed fragment is a triple $FT = (F, TF, ty)$, consisting of an instance fragment $F : Fr$, a type fragment $TF : TFr$, and fragment morphism $ty : F \to TF$ from instance to type. Set of typed fragments FrTy is defined as:*

$$FrTy = \{(F, TF, ty) \mid F \in Fr \wedge TF \in TFr \wedge ty \in F \to fr\ TF\}$$ □