Model Transformations as Free Constructions

Michael Löwe(✉)

FHDW Hannover, Freundallee 15, 30173 Hannover, Germany
michael.loewe@fhdw.de

Abstract. Hartmut Ehrig was an active researcher in Algebraic Specifications on the one hand and Graph and Model Transformations on the other hand. We demonstrate that these two research fields are closely connected, if we consider *generating* graph transformations only and use *partial algebras* instead of total algebras as the underlying category.

1 Introduction

In the last two decades, algebraic graph transformations became popular as a visual specification method for model transformations [8,9]. There are two major reasons for that: Graph transformations use a graphical notation and are rule-based. A graphical notation is adequate, since many modern modelling techniques are graphical themselves, like for example use case, class, state, or activity diagrams in the Unified Modelling Language UML [15] or process specifications in Business Process Model and Notation BPMN [14]. And a rule-based mechanism avoids explicit control structures. Thus, it opens up the chance for massively parallel transformations.

Model transformations define the mapping between artefacts in possibly different modelling languages. Examples are the mapping of Petri-Nets [20] to State Charts [17], entity relationship diagrams [4] to relational database schemata, XML-schemata [6] to UML class diagrams [15], or arbitrary UML class diagrams to UML class diagrams without symmetric associations and multiple inheritance[1]. In all these situations, model transformations are *generators*: given a model in the source language, the transformation process step by step generates a model in the target language together with a mapping that records the correspondence between the original items in the source to the generated items. Every model generation of this type must satisfy the following general requirements:

Termination. The generation process terminates for every finite source model.
Uniqueness. It produces a uniquely determined target for every source model.[2]
Persistence. The target generation does not change the source model.

These three requirements suggest that model transformations can be understood as some sort of (persistent) free construction from a suitable category of

[1] This transformation can serve as a prerequisite for the "compilation" of UML class diagrams to object-oriented programming languages like Java.

[2] Up to isomorphism, if the semantics of transformations is using category theory.

R. Heckel and G. Taentzer (Eds.): Ehrig Festschrift, LNCS 10800, pp. 142–159, 2018.
https://doi.org/10.1007/978-3-319-75396-6_8

source models to a suitable category of target models.[3] In this paper, we provide such an interpretation. For this purpose, we do not have to introduce any new definitions or results in terms of theorems and mathematical proofs. Instead, we interpret the available results in a slightly adjusted environment:

1. We restrict the well-known algebraic graph transformations to generating rules, i.e. to rules that do not delete or copy any items.
2. For the underlying category, we pass from total unary algebras[4] to arbitrary partial algebras.

The first adjustment leads to a special case of graph transformations, in which the three different algebraic approaches to graph transformation, namely the double-pushout [12], the single-pushout [18], and the sesqui-pushout approach [7] coincide. This is due to the fact that generating rules are simple morphisms $L \xrightarrow{r} R$. For rules of this format, the direct derivation at a match $m : L \to G$ is a simple pushout construction of r and m in *all three* algebraic approaches.

The second adjustment provides a simple encoding of a "growing" correspondence between items in the source model and generated items in the target model, namely by partial mappings that are getting more and more defined within the transformation process.[5] And, in partial algebras, the generation of elements, the creation of definedness for predicates, and the production of equivalences can be controlled by the *same* simple mechanism, namely Horn-type formulae.[6]

The rest of the paper is structured as follows. Section 2 introduces a typical example of a model transformation scenario, namely the generation of relational database schemata for object-oriented class diagrams. This example has also been used as a running example in [9].[7] We show by the example, that even complex model transformation tasks can be modelled by purely generating rules. Section 3 presents the rich theory of generating transformations in a very general categorical framework. Especially, we show in Sect. 3.2 that special generating transformation systems can be interpreted as specifications defining epi-reflective sub-categories of the underlying category. Section 4 presents the basic theory for algebraic specifications of partial algebras as a concrete syntax for transformation rules. In this framework, the sample transformation rules in Sect. 2 obtain a formal semantics, which allows a detailed review of all samples in Sect. 5. The analysis results in substantial improvements that, on the one hand, simplify the transformations from the practical point of view and, on the other hand, turn all of them into free constructions.

[3] For persistent free construction between abstract data types compare [5,10,11].
[4] Like graphs, hypergraphs, graph structures [18], or general functor categories from a finite category to the category of sets and mappings.
[5] Possibly without becoming total at last.
[6] Alfred Horn, American mathematician, 1918 – 2011.
[7] Compare [9], Example 3.6 on page 54.

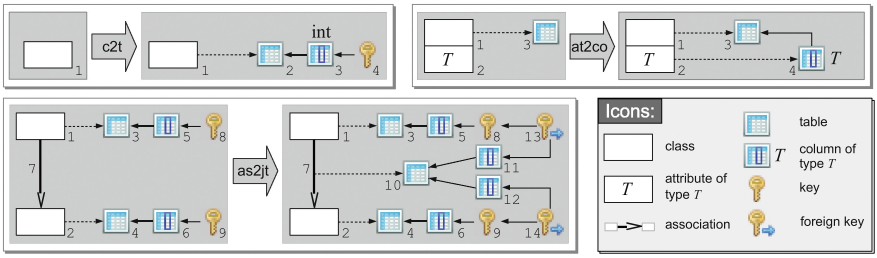

Fig. 1. Handling of classes, attributes, and associations in ST and CT

2 Sample Transformations – from Classes to Relations

A typical model transformation scenario is the generation of relational schemata for object-oriented class diagrams. In this section, we present transformation rules for all three patterns that are useful to cope with inheritance, namely Single Table Inheritance (ST), Class Table Inheritance (CT), and Concrete Class Table Inheritance (CCT) [13].[8] The presentation in this section stays on the intuitive level and trusts in the suggestive power of the rule visualisations as graphs with significant icons. Section 4 presents the necessary formal underpinning for a precise semantics which is described in Sect. 5.

In this section, we use the same mapping from class diagrams to relational schemata which has been used in Example 3.6 of [9]: Classes are mapped to tables, attributes are mapped to columns, and associations are mapped to junction tables. The corresponding three rules for Single Table Inheritance and Class Table Inheritance are depicted in Fig. 1.[9] For each class, the rule `c2t` generates a new table together with a column of numeric type (`int`) that is marked as key column[10]. That a class has been mapped is stored by a (partial) map indicated by the dotted arrow. The rule that generates columns in tables for class attributes is `at2co`. It is applicable to each attribute the class of which possesses an assigned table, possibly generated by an application of rule `c2t`. In order to simplify the presentation in this paper, we suppose that the set of base types (int, bool, string etc.) is identical in class models and relational schemata. Finally, associations between classes are mapped to junction tables by rule `as2jt`. This rule presupposes that the two classes at the ends of the association possess an

[8] This section uses material from [21].

[9] The correspondence of the items in the left hand side of the rule to items in the right-hand side is always indicated by the visual correspondence in the layouts. In Fig. 1, we give an additional explicit definition of the mapping from left to right by index numbers. In most cases, this explicit indication is superfluous. Since mappings must be type-conform, i.e. classes can only be mapped to classes, attributes can only be mapped to attributes etc., and the mapping must respect the graphical structure, the mapping is uniquely determined for all rules that are depicted below. Thus, we do not use the index numbers in the following.

[10] Standard key columns are `unique` and `not null`.

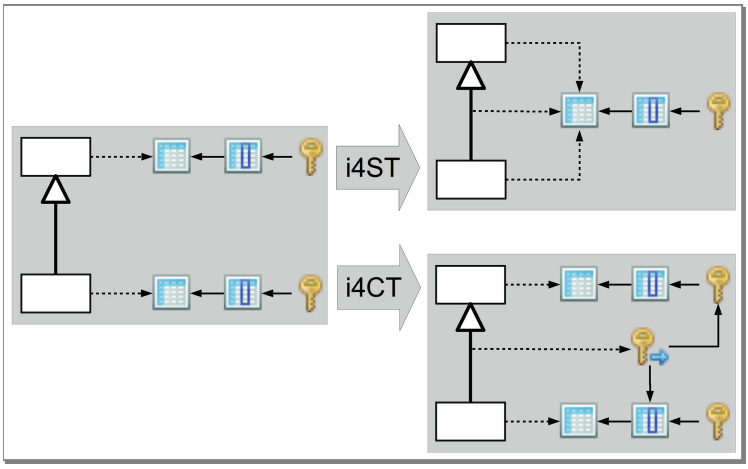

Fig. 2. Handling of inheritance in ST and CT

assigned table with a primary key column each, which have possibly been generated by two applications of rule `c2t`. To each association, it assigns (dotted arrow) a new table with two columns that both are marked as foreign keys.[11] The foreign keys reference the keys that have been found by the match. For the sake of simplicity, we suppose that the type of a foreign key column is implicitly set to the type of the referenced key column.

The handling of inheritance relations between classes is different in ST and CT. Figure 2 depicts the corresponding rules `i4ST` respectively `i4CT` which have the same left-hand side. The ST-pattern puts a complete inheritance hierarchy into a single table. This effect is implemented by the rule `i4ST` which merges the tables and keys for the super- and the sub-class of an inheritance relation and maps the relation itself to the same table, compare upper part of Fig. 2.[12,13]

Figure 3 depicts a sample transformation sequence for a small composite class diagram in the single-table scenario. In the first step, rule `c2t` is applied three times and produces a table with primary key column for each class in the start object. The second step ($2 \times$ `at4co`) handles the column generation for the two given attributes. The third step ($2 \times$ `i4ST`) merges the three tables that have been generated before and ensures that there is only one table for the complete inheritance hierarchy. Finally, the fourth step applies the rule `as2jt` once which adds the junction table for the given association.

The CT-pattern realises inheritance relations by foreign key references. Therefore, the rule `i4CT` maps an inheritance relation to a foreign key, compare lower part of Fig. 2. The key column of the table for the sub-class is

[11] Standard foreign key columns are `not null`.
[12] Merging of objects is expressed by non-injective rules.
[13] Due to the merging, some of the generated junction tables for association may no longer accurately model the association's semantics.

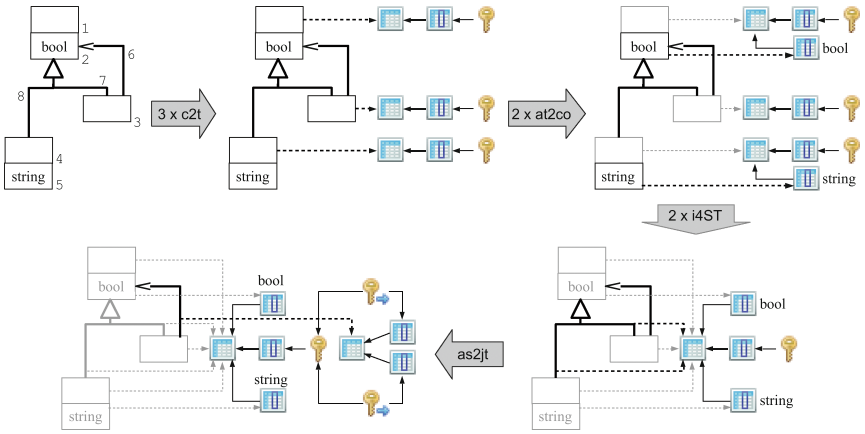

Fig. 3. Sample transformation sequence

simultaneously used as a foreign key reference to the key of the table for the super-class. This strategy requires coordinated key value generation for all tables "in an inheritance hierarchy" but provides unique object identity for all "parts" of the same object in different tables.

The CCT-pattern generates tables for all concrete classes only and "copies" all inherited attributes and associations to these tables. For the control of this copying process, we need a relation that provides all direct and indirect (transitive) sub-classes for a super-class. Figure 4 shows the rules, that "compute" the reflexive ($\mathtt{t}^0$) and transitive ($\mathtt{t}^*$) closure of the given inheritance relation ($\mathtt{t}^1$).

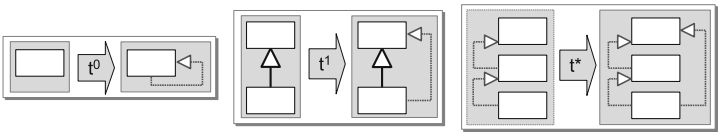

Fig. 4. Reflexive/transitive closure of inheritance

Since tables are generated for concrete classes only, we distinguish between abstract and concrete classes. Concrete classes are visualised by the annotation {c}. The rule cc2t in Fig. 5 generates tables and keys in the CCT-pattern.[14]

A single attribute can result in several columns, compare rule `cat2co` in Fig. 5: For an attribute a of type T in class c, a column of type T is generated into every table for a concrete class c' that is a sub-class of c. Thus, the attribute mapping gets indexed by the concrete sub-classes of the owner class

[14] The rule cc2t is a simple modification of rule `c2t` in Fig. 1. Here, the class-to-table mapping is partial, since abstract classes are never mapped in CCT.

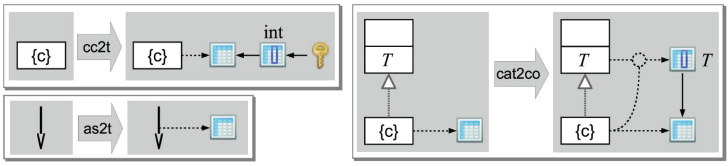

Fig. 5. Handling of classes, attributes, and associations in CCT

of the attribute, compare dotted circle in the visualisation of rule `cat2co` in Fig. 5.[15]

The handling of associations in CCT is even more complex. They cannot be mapped to *one* foreign key pair, since rows from several "unconnected" tables can be linked by instances of an association in this pattern. For an accurate mapping of the class model semantics, we need orthogonal combinations of foreign keys into the tables for all concrete sub-classes of the association's source class with foreign keys into the tables for all concrete sub-classes of the association's target class. The generation of these foreign keys is prepared by rule `as2t` in Fig. 5. It provides the table for all foreign key columns that are generated for an association.

The rules `s2co` and `t2co` in Fig. 6 generate these columns together with the foreign key references to the corresponding tables. These two rules are almost identical; `s2co` handles all concrete sub-classes of the association's source and `t2co` all concrete sub-classes of the association's target class. As in the case of the attribute mapping, the two mappings that store the correspondence between items in the class model and items in the relational schema are indexed by the concrete sub-class either on the source or the target side, compare the two dotted circles in Fig. 6.

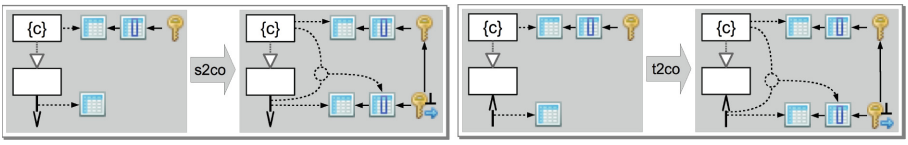

Fig. 6. Foreign keys for associations in CCT

For an accurate representation of the semantics of the class model, in each row of the "multi-junction table" exactly one foreign key on the source side and exactly one foreign key on the target side must be `not null`.[16] Thus, we need

[15] Rule `cat2co` also works for the own attributes of a concrete class due to the reflexivity rule $\mathtt{t}^0$ in Fig. 4. This situation requires non-injective matches!

[16] For the sake of simplicity of the presentation we do not generate a suitable `check` constraint in the relational schema.

foreign key columns that allow `null` values. This sort of foreign keys is depicted by a foreign key icon that is decorated by a $\perp$-symbol in Fig. 6.

3 Generating Transformations and Epi-Reflections

The samples in the preceding section show that model transformations can be specified by generating rules. This special case of transformation rules and transformation systems is presented in this section. We show that there is a rich theory especially wrt. parallel and sequential independence and parallel rule application. We assume an underlying category $\mathcal{C}$ with all small co-limits.[17,18]

3.1 Generating Transformation Systems

A *rule* is a morphism $r : L \to R$, a *match* for rule $r : L \to R$ in object G is a morphism $m : L \to G$, and a *direct transformation* with rule r at match m is defined by the pushout $(r\langle m\rangle : G \to r@m, m\langle r\rangle : R \to r@m)$ of the pair (r, m).[19] The derivation result is denoted by $r@m$, the morphisms $r\langle m\rangle$ is called the *trace* of the direct derivation, and the morphism $m\langle r\rangle$ the *co-match*.[20] The result $r@m$ is unique up to isomorphism, since pushouts are.

A *transformation system* $\mathbb{R}$ is a set of rules. The class $\mathbb{R}^{\rightarrow}$ of $\mathbb{R}$-*transformations* is the least class of morphisms which (i) contains all isomorphisms, (ii) contains all traces $r\langle m\rangle$ for all rules in $r \in \mathbb{R}$ and all suitable matches m, and (iii) is closed under composition. By $\mathbb{R}^{\rightarrow}_G = \{h \in \mathbb{R}^{\rightarrow} \mid \text{domain}(h) = G\}$, we denote the $\mathbb{R}$-transformations starting at object G. An object G is *final* wrt. a transformation system $\mathbb{R}$, if all $h \in \mathbb{R}^{\rightarrow}_G$ are isomorphisms.[21] A system $\mathbb{R}$ is

terminating if, for any infinite sequence $(t_i : G_i \to G_{i+1} \in \mathbb{R}^{\rightarrow})_{i\in\mathbb{N}}$, there is $n \in \mathbb{N}$, such that G_n is final,
confluent if, for any two $\mathbb{R}$-transformation $t_1 : G \to H_1$ and $t_2 : G \to H_2$, there are $\mathbb{R}$-transformations $t'_1 : H_2 \to K$ and $t'_2 : H_1 \to K$, and
functional if it is terminating and confluent.

Every generating transformation system is *strongly* confluent as the following argument demonstrates. Consider Fig. 7 which depicts two arbitrary direct

[17] A category has all small co-limits, if it has all co-limits for small diagram categories. A category is small if its collection of objects is a sets. As in [1], the family of morphisms in a category is a family of sets anyway.
[18] Examples for such categories are all total or partial Σ-algebras for a given signature Σ or every epi-reflective sub-category of such categories of Σ-algebras, see below.
[19] Neither rules nor matches are required to be monomorphisms. Rules and matches can be arbitrary morphisms.
[20] Sub-diagram (1) in Fig. 7 denotes a direct derivation with rule r_1 at match m_1.
[21] Object G being final wrt. system $\mathbb{R}$ does not mean that there are no matches for rules in $\mathbb{R}$ into G. But all these matches produce traces that are isomorphisms, i.e. do not have any effect.

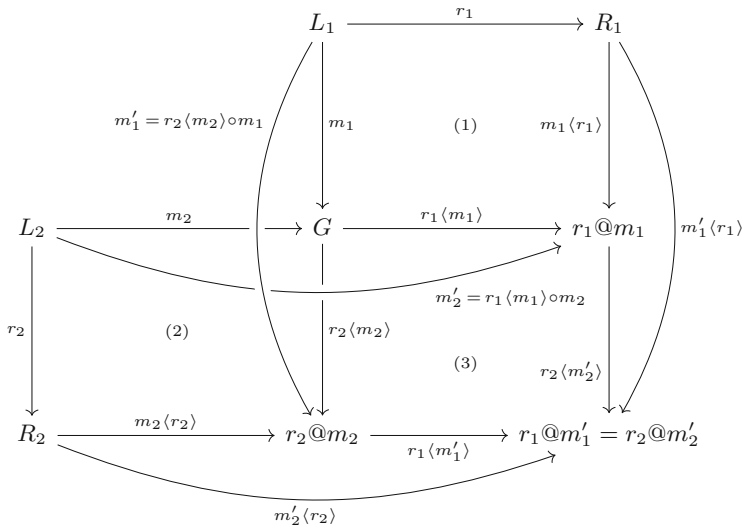

Fig. 7. Strong confluence

derivations with rules r_1 and r_2 at matches m_1 and m_2 respectively, i.e. sub-diagrams (1) and (2) are pushouts. Composing the original match of one rule with the trace induced by the other rule provides the two *residual* matches $m_1' = r_2\langle m_2\rangle \circ m_1$ and $m_2' = r_1\langle m_1\rangle \circ m_2$. Applying rule r_1 at that residual match m_1' leads to the pushout $(r_1\langle m_1'\rangle : r_2@m_2 \to r_1@m_1', m_1'\langle r_1\rangle : R_1 \to r_1@m_1')$. Since sub-diagram (1) is a pushout and $(r_1\langle m_1'\rangle \circ r_2\langle m_2\rangle) \circ m_1 = r_1\langle m_1'\rangle \circ m_1' = m_1'\langle r_1\rangle \circ r_1$, we obtain the unique morphism from $r_1@m_1$ to $r_1@m_1'$, which we call $r_2\langle m_2'\rangle$, satisfying $r_2\langle m_2'\rangle \circ m_1\langle r_1\rangle = m_1'\langle r_1\rangle$ and $r_2\langle m_2'\rangle \circ r_1\langle m_1\rangle = r_1\langle m_1'\rangle \circ r_2\langle m_2\rangle$. Since pushouts decompose, the sub-diagram (3) in Fig. 7 is a pushout, i.e. the pair $(r_2\langle m_2'\rangle, r_1\langle m_1'\rangle)$ is pushout of $(r_1\langle m_1\rangle, r_2\langle m_2\rangle)$. Since pushouts compose, the sub-diagrams (2) and (3) together constitute a pushout, i.e. the pair $(r_1\langle m_1'\rangle \circ m_2\langle r_2\rangle, r_2\langle m_2'\rangle)$ is pushout of r_2 and $r_1\langle m_1\rangle \circ m_2 = m_2'$. Therefore $r_2\langle m_2'\rangle$ is the trace of the direct derivation with rule r_2 at match m_2' and $r_1\langle m_1'\rangle \circ m_2\langle r_2\rangle$ is the co-match $m_2'\langle r_2\rangle$. Combining these results, we obtain:

$$r_1@(r_2\langle m_2\rangle \circ m_1) = r_2@(r_1\langle m_1\rangle \circ m_2) \text{ and}$$
$$r_2\langle m_2'\rangle \circ r_1\langle m_1\rangle = r_1\langle m_1'\rangle \circ r_2\langle m_2\rangle.$$

Thus, the system is strongly confluent, which implies that it is confluent and, furthermore, that *it is functional, iff it is terminating*. That every generating transformation system is strongly confluent has some further direct positive consequences, especially wrt. sequential independence and parallel transformations.

Two composable traces $r_2\langle m_2\rangle \circ r_1\langle m_1\rangle$ are *sequentially independent*, if there is a match m_2' for the rule r_2 such that $m_2 = r_1\langle m_1\rangle \circ m_2'$. In the case of sequential independence, the order of rule application can be interchanged,

Algorithm 1. Calculation of final objects

(1) Set the current object c to the start object o.
(2) Find all matches for all $\mathbb{R}$-rules in c
(3) Produce trace $t : c \to c'$ for corresponding parallel rule at induced parallel match.
(4) If t is an isomorphism stop and return c.
(5) Otherwise set the current object c to c' and continue at (2).

i.e. $r_2\langle m_2\rangle \circ r_1\langle m_1\rangle = r_1\langle r_2\langle m_2'\rangle \circ m_1\rangle \circ r_2\langle m_2'\rangle$. The proof for this result is a simple reduction to strong confluence, compare Fig. 7.

Given two rules $r_1 : L_1 \to R_1$ and $r_2 : L_2 \to R_2$, we can construct the *parallel rule* $r_1{+}r_2 : L_1{+}L_2 \to R_1{+}R_2$ as the unique co-product morphism, where $(i_{L_1} : L_1 \to L_1{+}L_2,\ i_{L_2} : L_2 \to L_1{+}L_2)$ is the co-product of the rule's left-hand sides, $(i_{R_1} : R_1 \to R_1{+}R_2,\ i_{R_2} : R_2 \to R_1{+}R_2)$ is the co-product of the rule's right-hand sides, and $r_1{+}r_2$ satisfies $i_{R_1} \circ (r_1 + r_2) = r_1 \circ i_{L_1}$ and $i_{R_2} \circ (r_1 + r_2) = r_2 \circ i_{L_2}$. Having two matches m_1 and m_2 in the same object for r_1 and r_2 respectively, the parallel match $m_1{+}m_2$ is uniquely determined by $(m_1 + m_2) \circ i_{L_1} = m_1$ and $(m_1 + m_2) \circ i_{L_2} = m_2$. Since any co-limit construction of the same situation in any order results in the same object (up to isomorphism), we immediately obtain: $r_1{+}r_2\langle m_1 + m_2\rangle = r_2\langle r_1\langle m_1\rangle \circ m_2\rangle \circ r_1\langle m_1\rangle = r_1\langle r_2\langle m_2\rangle \circ m_1\rangle \circ r_2\langle m_2\rangle$.

This result provides a good (maximal parallel) strategy for the search of a final object for a start object o in a system $\mathbb{R}$, compare Algorithm 1. If the system is terminating, the algorithm finds *the* final object for all start objects. Unfortunately, most systems are not terminating as the following argument shows.

The application of a rule r at a match m is idempotent, i.e. $r\langle r\langle m\rangle \circ m\rangle$ is an isomorphism, if and only if the trace $r\langle m\rangle$ is an epimorphism. For the proof, consider Fig. 7 again and let $r_1 = r_2$ as well as $m_1 = m_2$. For the if-part, let $r_1\langle m_1\rangle$ as well as $r_2\langle m_2\rangle$ be epic. Then $r_1\langle m_1'\rangle$, and $r_2\langle m_2'\rangle$ are epic, since pushouts preserve epimorphisms. Since pushouts are unique up to isomorphism, there is an isomorphism $i : r_1@m_1 \to r_2@m_2$ such that $i \circ r_1\langle m_1\rangle = r_2\langle m_2\rangle$. Since (3) is pushout, we obtain $j : r_1@m_1' \to r_2@m_2$ with $j \circ r_1\langle m_1'\rangle = \mathrm{id}$ and $j \circ r_2\langle m_2'\rangle = i$. Thus, $r_1\langle m_1'\rangle$ is section and epic, which means that it is isomorphism. For the only-if-part, suppose the application is idempotent, i.e. $r_1\langle m_1'\rangle$ is isomorphism and $r_2\langle m_2'\rangle$ is isomorphism with inverse morphism j. Then $r_1\langle m_1\rangle$ is epic: $h \circ r_1\langle m_1\rangle = k \circ r_1\langle m_1\rangle$ implies $h \circ r_1\langle m_1\rangle = k \circ r_1\langle m_1\rangle = k \circ j \circ r_2\langle m_2'\rangle \circ r_1\langle m_1\rangle = (k \circ j \circ r_1\langle m_1'\rangle) \circ r_2\langle m_2\rangle$. Since $(r_1\langle m_1'\rangle, r_2\langle m_2'\rangle)$ is pushout, there is unique morphism u such that $u \circ r_1\langle m_1'\rangle = k \circ j \circ r_1\langle m_1'\rangle$ and, since $r_1\langle m_1'\rangle$ is isomorphism, (i) $u = k \circ j$ as well as $u \circ r_2\langle m_2'\rangle = h$ and, since j is inverse of $r_2\langle m_2'\rangle$, (ii) $h \circ j = u \circ r_2\langle m_2'\rangle \circ j = u$. Now (i) and (ii) imply that $h \circ j = k \circ j$ and $h = k$ since j is isomorphism.

Therefore, a single non-epic trace $r\langle m\rangle$ in a system prevents termination, since the rule r can be applied over and over again at residuals of m with "new results". Thus, a necessary condition for termination is that all traces in the system are epic. Since pushouts preserve epimorphisms, this property can be guaranteed, if we restrict rules to epimorphisms. Such systems consisting of epic rules only possess another important property as the next sub-section demonstrates.

3.2 Transformation Systems as Epi-Reflections

Every generating transformation system $\mathbb{R}$ in the sense of Sect. 3.1 which consists of epic rules only, can be interpreted as a specification of an epi-reflective sub-category of the underlying category $\mathcal{C}$. This section recapitulates the construction of this epi-reflection.[22]

Every epic transformation rule $r : L \twoheadrightarrow R$ can be interpreted as a *constructive axiom.*[23] It is *finite* or of *Horn-type*, if it satisfies the following condition: For every chain of morphisms $(m_i : O_i \to O_{i+1})_{i\in\mathbb{N}}$ with the co-limit $\big(C, (c_i : O_i \to C)_{i\in\mathbb{N}}\big)$, every morphism $p : L \to C$ into the co-limit object factors through an object in the chain, i.e. there is $i \in \mathbb{N}$ and a morphism $p_i : L \to O_i$ with $c_i \circ p_i = p$.

A morphism $m : L \to A$ *solves* axiom $r : L \twoheadrightarrow R$ in object A, written $m \models r$, if there is morphism $m^r : R \to A$ such that $m^r \circ r = m$.[24] An object A *satisfies* axiom r, written $A \models r$, if every morphism $m : L \to A$ solves r. An object A *satisfies* a transformation system $\mathbb{R}$ of epic rules, written $A \models \mathbb{R}$, if $A \models r$ for all $r \in \mathbb{R}$. The full sub-category of $\mathcal{C}$ which contains all objects satisfying $\mathbb{R}$ is denoted by $\mathcal{C}_{\mathbb{R}}$. Every such sub-category turns out to be an epi-reflection of $\mathcal{C}$.

Given an object A and a transformation system $\mathbb{R}$ of epic rules,

$$A_{\mathbb{R}} = \left\{ A \xleftarrow{m} L_m \xrightarrow{r_m} R_m \mid (r : L \twoheadrightarrow R) \in \mathbb{R}, m : L \to A \right\}$$

denotes the diagram of all occurrences of the left-hand sides of all rules in the transformation system in A.[25] In that diagram, for every morphism $m : L \to A$, the morphism $r_m : L_m \to R_m$ is a "copy" of $r : L \twoheadrightarrow R$. The co-limit of $A_{\mathbb{R}}$ is denoted by $A^{\mathbb{R}} = \left(A^{\mathbb{R}}, r^*_A : A \to A^{\mathbb{R}}, \big(m^{r_m} : R_m \to A^{\mathbb{R}}\big)_{\left(A \xleftarrow{m} L_m \xrightarrow{r_m} R_m\right) \in A_{\mathbb{R}}}\right)$ and we have $r^*_A \circ m = m^{r_m} \circ r_m$ for all $\left(A \xleftarrow{m} L_m \xrightarrow{r_m} R_m\right) \in A_{\mathbb{R}}$. The morphism r^*_A is an epimorphism, since the equation (i) $h \circ r^*_A = k \circ r^*_A$ implies, for all $\left(A \xleftarrow{m} L_m \xrightarrow{r_m} R_m\right) \in A_{\mathbb{R}}$, $h \circ m^{r_m} \circ r_m = h \circ r^*_A \circ m = k \circ r^*_A \circ m = k \circ m^{r_m} \circ r_m$ and, since rules are epimorphisms, (ii) $h \circ m^{r_m} = k \circ m^{r_m}$. Since $A^{\mathbb{R}}$ is co-limit, (i) and (ii) imply $h = k$.

For given object A and transformation system $\mathbb{R}$, consider $(r^*_{A_i} : A_i \twoheadrightarrow A_{i+1})_{i\in\mathbb{N}}$ as the chain of epimorphisms starting at $A_1 = A$ and having $A_{i+1} = A_i^{\mathbb{R}}$. We denote the co-limit of this chain by $\big(\mathbb{R}(A), (a_i : A_i \twoheadrightarrow \mathbb{R}(A))_{i\in\mathbb{N}}\big)$ and obtain $\big(a_{i+1} \circ r^*_{A_i} = a_i\big)_{i\in\mathbb{N}}$. All these co-limit morphism are epimorphisms, since all morphisms in the chain are epic.

Furthermore $\mathbb{R}(A) \in \mathcal{C}_{\mathbb{R}}$: Let $(r : L \twoheadrightarrow R) \in \mathbb{R}$ and let $m : L \to \mathbb{R}(A)$. Since r is finite, there is $i \in \mathbb{N}$ and $m_i : L \to A_i$ with $a_i \circ m_i = m$. Since $A_{i+1} = A_i^{\mathbb{R}}$, we obtain morphism $m^r_i : R \to A_{i+1}$ with $m^r_i \circ r = r^*_{A_i} \circ m_i$. Putting all parts

[22] The principle set-up follows [1], page 278 ff.

[23] In this context, the objects L and R are called *premise* and *conclusion* respectively.

[24] The morphism m^r is unique, if it exists, since r is epic.

[25] $A_{\mathbb{R}}$ is a small diagram, since the family of morphisms in a category is a family of *sets*, compare definition of categories in [1].

together provides: $(a_{i+1} \circ m_i^r) \circ r = a_{i+1} \circ r^*_{A_i} \circ m_i = a_i \circ m_i = m$, such that we found $a_{i+1} \circ m_i^r$ as the desired morphism $m^r : R \to \mathbb{R}(A)$ with $m^r \circ r = m$.

Finally, we get the following result*: For a transformation system* $\mathbb{R}$ *and object* A, $a_1 : A \twoheadrightarrow \mathbb{R}(A)$ *is the epi-reflector for* A *into* $\mathcal{C}_\mathbb{R}$. The proof is straightforward: If $X \in \mathcal{C}_\mathbb{R}$ and $f : A \to X$ are given, we show by induction on i that there are morphisms $(f_i : A_i \to X)_{i\in\mathbb{N}}$ for all $(A_i)_{i\in\mathbb{N}}$ in the chain $\left(r^*_{A_i} : A_i \twoheadrightarrow A_i^\mathbb{R}\right)_{i\in\mathbb{N}}$ constructed above. Since $A_1 = A$, the induction can start with $f_1 : A_1 \to X := f : A \to X$. Now let, as induction hypothesis, $f_i : A_i \to X$ be given. By construction $A_{i+1} = A_i^\mathbb{R}$ and $A_i^\mathbb{R}$ is a co-limit object. Let $(r : L \twoheadrightarrow R) \in \mathbb{R}$ and $m : L \to A_i$ be a morphism to A_i, then $f_i \circ m : L \to X$ is a morphism to X. Since $X \models \mathbb{R}$, there is $(f_i \circ m)^r : R \to X$ with $(f_i \circ m)^r \circ r = f_i \circ m$. Thus, f_i together with the family of these morphisms are a co-cone for the diagram that has been used to construct $A_{i+1} = A_i^\mathbb{R}$ from A_i. Since $A_i^\mathbb{R}$ is the co-limit of this diagram, we obtain the unique morphism $f_{i+1} : A_i^\mathbb{R} \to X$ that satisfies $f_{i+1} \circ r^*_{A_i} = f_i$. This completes the induction. Now $\left(X, (f_i : A_i \to X)_{i\in\mathbb{N}}\right)$ is a co-cone for the chain $\left(r^*_{A_i} : A_i \twoheadrightarrow A_i^\mathbb{R}\right)_{i\in\mathbb{N}}$. Since $\mathbb{R}(A)$ is the limit of this chain, we get a morphism $f^* : \mathbb{R}(A) \to X$ that satisfies $(f^* \circ a_i = f_i)_{i\in\mathbb{N}}$ and, especially, $f^* \circ a_1 = f_1 = f$. Uniqueness of f^* follows from a_1 being epic.

The co-limit construction for the chain $\left(r^*_{A_i} : A_i \twoheadrightarrow A_{i+1}\right)_{i\in\mathbb{N}}$ is superfluous, if $r^*_{A_k}$ is an isomorphism for some $k \in \mathbb{N}$. In this case, (i) $A_k \in \mathcal{C}_\mathbb{R}$, (ii) r^*_{k+j} is isomorphism for all $j \in \mathbb{N}$, (iii) A_{k+1} is the limit of the chain, and (iv) the reflector for A is given by $r^*_{A_K} \circ \cdots \circ r^*_{A_1} : A \twoheadrightarrow A_{k+1}$.

The presented approximation of the epi-reflector for a transformation system $\mathbb{R}$ with epic rules is an instance of Algorithm 1. Each approximation step $r^*_A : A \twoheadrightarrow A^\mathbb{R}$ is the trace of an application of a maximal parallel rule, compare diagram $A_\mathbb{R}$ above. And the approximation stops after finitely many steps, if and only if the computed object is final wrt. $\mathbb{R}$, i.e. admits isomorphic $\mathbb{R}$-transformations only. Thus, Algorithm 1 calculates the epi-reflector in $\mathcal{C}_\mathbb{R}$ for every start object o in a terminating transformation system $\mathbb{R}$ with epic rules.

4 Partial Algebras

Section 3.2 shows that all generating transformation systems in which all rules are epimorphisms induce free constructions and possess useful properties like idempotent rule applications that facilitates the analysis with respect to termination. Unfortunately, all epimorphisms in categories of *total* algebras, which usually constitute the underlying category for most (graph) transformation frameworks, are surjective, i.e. are just able to generate new equalities. Therefore, frameworks based on total algebras must live with non-epic rules and need an additional machinery for termination, like *negative application conditions* [16], *source consistence derivations* in triple graph grammars [2], or artificial *translation attributes* as in [9], Sect. 7.4. The situation stays simple, if we pass from total to *partial* algebras [3,19] as we recapitulate in this section.

A *signature* $\Sigma = (S, O)$ consists of a set of sorts S and a family of operations $O = (O_{d,c})_{d,c\in S^*}$. For $d, c \in S^*$ and $f \in O_{d,c}$, d is called the *domain*

specification of f and c is called the *co-domain specification*. An (*algebraic*) *system* $A = \left(A_S, O^A\right)$ for a given signature $\Sigma = (S, O)$ consists of a family $A_S = (A_s)_{s \in S}$ of *carrier sets*, and a family $O^A = \left(f^A : A^d \to A^c\right)_{f \in O_{d,c}, d, c \in S^*}$ of *partial functions*.[26]

This set-up allows functions that provide several results simultaneously, since co-domain specifications are taken from the free monoid over the sort set. Especially, operations with an empty co-domain specification are possible. They are interpreted as *predicates* in algebraic systems: If $p \in O_{d,\varepsilon}$ for $d \in S^*$, the function $p^A : A^d \to \{*\}$ maps to the one-element-set in every system A. Thus p^A singles out the elements in A^d for which it is defined, i.e. for which it is "true".

Given two systems A and B with respect to the same signature $\Sigma = (S, O)$, a *homomorphism* $h : A \to B$, is given by a family of *total* mappings $h = (h_s : A_s \to B_s)_{s \in S}$ such that the following condition is satisfied:[27]

$$\forall d, c \in S^*, f \in O_{d,c}, x \in A^d \colon \text{if } f^A(x) \text{ is defined, } f^B\left(h^d(x)\right) = h^c\left(f^A(x)\right). \quad (1)$$

The condition (1) means for the special case where $f \in O_{d,\epsilon}$, that f^B must be defined for $h^d(x)$, if f^A is defined for x. The concrete value of these functions is irrelevant, since there is a single value in $\{*\}$ and $h^\varepsilon = \mathrm{id}_{\{*\}}$. By $\mathsf{Sys}(\Sigma)$, we denote the category of all Σ-systems and all Σ-homomorphisms. For every signature Σ, $\mathsf{Sys}(\Sigma)$ has all small limits and co-limits for each signature Σ.[28]

The most important property of $\mathsf{Sys}(\Sigma)$ for the purposes of this paper is, that epimorphisms are not necessarily surjective. This fact can be demonstrated by a simple example using the signature with one sort $\mathtt{N}$ and one unary operation $\mathtt{s} \in O_{\mathtt{N},\mathtt{N}}$, the system $K = \left(K_\mathtt{N} = \{x\}, \mathtt{s}^K = \emptyset\right)$, the system of natural numbers $N = \left(N_\mathtt{N} = \mathbb{N}_0, \mathtt{s}^N :: i \mapsto i + 1\right)$, and the morphism $k : K \to N$ defined by $x \mapsto 0$. This morphism is epic, since N is generated by the function $\mathtt{s}^N$ starting at value $0 = k(x)$.[29] However, the morphism $k' : K \to N$ with $x \mapsto 1$ is not epic, since the value 0 is not reachable by function calls of $\mathtt{s}^N$ starting at 1.[30]

For a general characterisation of epimorphisms in $\mathsf{Sys}(\Sigma = (S, O))$, we need the following closure operations for a family of subsets $(B_s \subseteq A_s)_{s \in S}$ of a system A. For all sorts $s \in S$:

$$B_s^0 = B_s \cup \left\{ y \in A_s \;::\; f^A(*) = (p, y, q),\ f \in O_{\epsilon,v} \right\}$$

$$B_s^{i+1} = B_s^i \cup \left\{ y \in A_s \;::\; f^A(x) = (p, y, q),\ f \in O_{w,v},\ x \in \left(B^i\right)^w,\ |w| \geq 1 \right\}$$

$$B_s^* = \bigcup_{i \in \mathbb{N}_0} B_s^i$$

[26] Given a family of sets $A = (A_s)_{s \in S}$, $k \geq 0$, and $s_1 \ldots s_k \in S^*$, $A^{s_1 \cdots s_k} = A_{s_1} \times \cdots \times A_{s_k}$.

[27] For a family of mappings $f = (f_s : A_s \to B_s)_{s \in S}$, $k \geq 0$, and $w = s_1 \ldots s_k \in S^*$, $f^w : A^w \to B^w$ is defined by $f^w(x_1, \ldots, x_k) = (f_{s_1}(x_1), \ldots, f_{s_k}(x_k))$ for all $(x_1, \ldots x_k) \in A^w$.

[28] Compare [19].

[29] Any two morphisms $p, q : N \to M$ with $p(k(x)) = q(k(x))$ must coincide on all natural numbers due to the homomorphism condition (1).

[30] It is easy to construct $p, q : N \to M$ with $p(k'(x)) = q(k'(x))$ that differ at value 0.

A Σ -morphism $h : A \to B$ is epic, if and only if $h(A)^*_s = B_s$ for all sorts $s \in S$, i.e. if and only if the system B is function-generated starting at the h -image of A in B.[31] Therefore, a constructive axiom in the sense of Sect. 3.2 in a category $\mathsf{Sys}(\Sigma)$ of algebraic systems can generate new elements in the carriers (as long as they are operation generated), can generate new "truths" by defining new predicate instances, and can generate new equalities if it is not injective.

Constructive axioms are usually presented as finite implications, the elementary building blocks of which are formulae. Given signature $\Sigma = (S, O)$ and variable set $X = (X_s)_{s\in S}$, the set of formulae $\mathcal{F}^{\Sigma,X} = \left(T_s^{\Sigma,X}\right)_{s\in S} \cup F^{\Sigma,X}$ is defined by:

$$x \in T_s^{\Sigma,X} \text{ if } x \in X_s \tag{2}$$
$$f_i(t) \in T_{s_i}^{\Sigma,X} \text{ if } f \in O_{w,s_1\dots s_k},\, t \in \left(T^{\Sigma,X}\right)^w, 1 \le i \le k, k \ge 1 \tag{3}$$
$$f(t) \in F^{\Sigma,X} \text{ if } f \in O_{w,\epsilon},\, t \in \left(T^{\Sigma,X}\right)^w \tag{4}$$
$$l = r \in F^{\Sigma,X} \text{ if } l, r \in T_s^{\Sigma,X}, s \in S \tag{5}$$

A syntactical presentation $P_X = \left(X, P \subseteq \mathcal{F}^{\Sigma,X}\right)$ of a Σ-system consists of a variable set X and a set of formulae P; it is *finite*, if X and P are finite sets. The *presented system* $\mathbf{A}^{P_X} = \mathbf{T}^{P_X}/_{\equiv}$ is constructed as follows. The *carriers* $\left(\mathbf{T}_s^{P_X}\right)_{s\in S}$ are inductively defined by:

$$x \in \mathbf{T}_s^{P_X} \text{ if } x \in X_s \text{ or } \left(x \in P \text{ and } x \in T_s^{\Sigma,X}\right) \tag{6}$$
$$f_j(x) \in \mathbf{T}_{s_j}^{P_X} \text{ if } f_i(x) \in \mathbf{T}_{s_i}^{P_X}, f \in O_{w,s_1\dots s_k}, 1 \le i, j \le k \tag{7}$$
$$t_j \in \mathbf{T}_{s_j}^{P_X} \text{ if } f_i(t_1, \dots t_m) \in \mathbf{T}_{s'_i}^{P_X}, f \in O_{s_1\dots s_m, s'_1\dots s'_n}, 1 \le j \le m, i \le n \tag{8}$$
$$t_j \in \mathbf{T}_{s_j}^{P_X} \text{ if } p(t_1, \dots t_k) \in P, p \in O_{s_1\dots s_k,\epsilon}, 1 \le j \le k, k \ge 1 \tag{9}$$
$$l, r \in \mathbf{T}_s^{P_X} \text{ if } (l = r) \in P \text{ and } l, r \in T_s^{\Sigma,X} \tag{10}$$

For an *operation* $f \in O_{w,s_1\dots s_k}$ $(k \ge 1)$ and a possible argument $x \in \left(\mathbf{T}^{P_X}\right)^w$, $f^{\mathbf{T}^{P_X}}(x) = (f_1(x), \dots, f_k(x))$, if $f_j(x) \in \mathbf{T}_{s_j}^{P_X}$ for all $1 \le j \le k$. For a *predicate* $p \in O_{w,\epsilon}$ and a possible argument $x \in \left(\mathbf{T}^{P_X}\right)^w$, $p^{\mathbf{T}^{P_X}}(x)$ is defined, if $p(x) \in P$. And the *quotient* relation $\equiv\, \subseteq \mathbf{T}^{P_X} \times \mathbf{T}^{P_X}$ is the smallest congruence containing $\{(l, r) \mid (l = r) \in P\}$.[32]

Since $\mathbf{T}^{P_X}$ is closed wrt. sub-terms, compare Eq. (8), and $\equiv$: $\mathbf{T}^{P_X} \to \mathbf{A}^{P_X}$ is surjective, $\mathbf{A}^{P_X}$ is generated by X which means, that there is an epimorphism $x^{P_X} : X \twoheadrightarrow \mathbf{A}^{P_X}$ mapping x to $[x]_{\equiv}$.[33] If we have two syntactical presentations $P_X = \left(X, P \subseteq \mathcal{F}^{\Sigma,X}\right)$ and $C_X = \left(X, C \subseteq \mathcal{F}^{\Sigma,X}\right)$ with the same variable set X

[31] Here, $h(A) = (h_s(A_s))_{s\in S}$ and $h_s(A_s) = \{y \in B_s \mid y = h_s(x), x \in A_s\}$. For a proof of the proposition, compare [19].

[32] An equivalence $\equiv$ on a Σ-System A is a congruence, if $f^A(x_1, \dots x_m) = (y_1, \dots y_n)$, $f^A(x'_1, \dots x'_m) = (y'_1, \dots y'_n)$ and $x_i \equiv x'_i$ for all $1 \le i \le m$ implies $y_j \equiv y'_j$ for all $1 \le j \le n$ for all operations $f \in O$.

[33] Every variable set is a Σ-system with completely undefined operations!

Specification 1. Formalisation of class models

```
CM:= sorts C(lass), B(asetype), At(tribute), As(sociation), I(nheritance)
     opns     int:     ⟶B,      conc:C
           o(wner):At⟶C, t(arget):At⟶B
           o(wner):As⟶C, t(arget):As⟶C
               sub: I⟶C,  sup(er): I⟶C,
                                <=: C,C
     axms       x:C::==> <=(x,x)                                        (a1)
           x:I;l,u:C::sub(x)=l,sup(x)=u ==> <=(l,u)                     (a2)
             x,y,z:C::    <=(x,y),<=(y,z) ==> <=(x,z)                   (a3)
               x,y:C::    <=(x,y),<=(y,x) ==> x=y                       (a4)
```

such that $P \subseteq C$, then the kernel of x^{Px} is contained in the kernel of x^{Cx} and we obtain an epimorphism $\overrightarrow{\overrightarrow{P_XC}} : \mathbf{A}^{Px} \twoheadrightarrow \mathbf{A}^{Cx}$ with $\overrightarrow{\overrightarrow{P_XC}} \circ x^{Px} = x^{Cx}$.[34]

A *Horn-type presentation* $H = \left(X, P \subseteq C \subseteq \mathcal{F}^{\Sigma,X}\right)$ of an axiom consists of a finite variable set X, a finite syntactical presentation $C \subseteq \mathcal{F}^{\Sigma,X}$, called the *conclusion*, and a sub-presentation P of C, called the *premise*. The *presented axiom* is the uniquely determined epimorphism $\overrightarrow{\overrightarrow{P_XC}}$. A system *satisfies* H, if it satisfies $\overrightarrow{\overrightarrow{P_XC}}$, compare Sect. 3.2.

Specification 1 presents a formalisation `CM` of the class diagrams that we used in Sect. 2, i. e. class diagrams are `CM`-algebras. As an example, consider the class diagram which is the start object in Fig. 3. It is modelled by the following `CM`-algebra S: $S_{\mathtt{C}} = \{1,3,4\}$; $S_{\mathtt{B}} = \{\mathtt{int},\mathtt{bool},\mathtt{string}\}$; $S_{\mathtt{At}} = \{2,5\}$; $S_{\mathtt{As}} = \{6\}$; $S_{\mathtt{I}} = \{7,8\}$; $\mathtt{int}^S :: * \mapsto \mathtt{int}$; $\mathtt{conc}^S = \{1,3,4\}$; $\mathtt{o}^S_{\mathtt{At}} :: 2 \mapsto 1, 5 \mapsto 4$; $\mathtt{t}^S_{\mathtt{At}} :: 2 \mapsto \mathtt{bool}, 5 \mapsto \mathtt{string}$; $\mathtt{o}^S_{\mathtt{As}} :: 6 \mapsto 3$; $\mathtt{t}^S_{\mathtt{As}} :: 6 \mapsto 1$; $\mathtt{sub}^S :: 7 \mapsto 3, 8 \mapsto 4$; $\mathtt{sup}^S :: 7 \mapsto 1, 8 \mapsto 1$; $\mathtt{<=}^S = \{(1,1),(3,3),(4,4),(3,1),(4,1)\}$.

All underlined operation are implicitly required to be total. This requirement can be explicitly specified by very simple axioms. For example, the axiom for the operation `o(wner):At⟶C` is: `x:At :: ==> o(x)`. The axioms `(a1)` – `(a4)` specify that inheritance is hierarchical, i.e. induces a partial order `<=`. The rules $\mathtt{t}^0$, $\mathtt{t}^1$, and $\mathtt{t}^*$ in Fig. 4 are picturesque visualisations of the epimorphisms presented by axioms `(a1)`, `(a2)`, and `(a3)` respectively.

Specification 2. Formalisation of relational schemata

```
RS:= sorts T(able), B(asetype), Co(lumn), K(ey), F(oreign)K(ey)
     opns       int:   ⟶B
           ta(ble):Co⟶T,    t(ype):Co⟶B,   n(ullable):Co
           c(olumn):K ⟶Co, c(olumn):FK⟶C,   r(efers):FK⟶K
```

[34] Compare for example Theorem 99 (Homomorphism Theorem 1) in [19].

Specification 3. Formal basis for Single Table Inheritance (ST)

ST:= CM +Basetype RS +	ST' := CM +Basetype RS +
opns C2T: Class ⟶ Table	opns C2K: Class ⟶ Key
At2Co: Attribute ⟶ Column	At2Co: Attribute ⟶ Column
As2JT: Association ⟶ Table	As2FKP: Association ⟶ FK,FK
I2T: Inheritance ⟶ Table	I2K: Inheritance ⟶ Key

5 Sample Transformations – Revisited

The informally introduced model transformation rules in Sect. 2 can be precisely formalised on the basis of the definitions in Sect. 4. Specifications 1 and 2 specify class models and relational schemata respectively. For the model transformation pattern "Single Table Inheritance (ST)", we devise the partial operations depicted in the specification ST in the left part of Specification 3.[35] With this interpretation all rules in Fig. 1 and the rule i4ST in Fig. 2 depict morphisms in Sys(ST). As an example, consider the rule c2t in Fig. 1. Its left-hand side L is the following ST-algebra: $L_{\mathtt{C}} = \{1\}$; $L_{\mathtt{B}} = \{\mathtt{int},\mathtt{bool},\mathtt{string}\}$; $\mathtt{int}^L :: * \mapsto \mathtt{int}$; $\mathtt{conc}^L = \{1\}$; $\mathtt{<=}^L = \{(1,1)\}$; and all other components of L are empty. Its right-hand side is represented by the ST-algebra R: $R_{\mathtt{C}} = \{1\}$; $R_{\mathtt{B}} = \{\mathtt{int},\mathtt{bool},\mathtt{string}\}$; $\mathtt{int}^R :: * \mapsto \mathtt{int}$; $\mathtt{conc}^R = \{1\}$; $\mathtt{<=}^R = \{(1,1)\}$; $R_{\mathtt{T}} = \{2\}$; $R_{\mathtt{Co}} = \{3\}$; $R_{\mathtt{K}} = \{4\}$; $\mathtt{ta}^L :: 3 \mapsto 2$; $\mathtt{t}^L_{\mathtt{Co}} :: 3 \mapsto \mathtt{int}$; $\mathtt{c}^L_{\mathtt{K}} :: 4 \mapsto 3$; $\mathtt{C2T}^L :: 1 \mapsto 2$; and all other components of R are empty. The rule morphism $r : L \to R$ maps as follows: $r_{\mathtt{C}} :: 1 \mapsto 1$; $r_{\mathtt{B}} :: \mathtt{int} \mapsto \mathtt{int}, \mathtt{bool} \mapsto \mathtt{bool}, \mathtt{string} \mapsto \mathtt{string}$; and all other components of the morphism are empty.

Unfortunately, the corresponding transformation system is not terminating for almost all class models. This is due to the fact, that the rules c2t and as2jt are not epic and induce non-epic traces. Therefore, they are not idempotent.

This defect can be avoided by a simple reengineering of ST to the specification ST' in the right part of Specification 3 together with the adapted rules in Fig. 8 and the adapted rule i4ST' in Fig. 9 which are epimorphism and guarantee termination of the transformation system for finite class models, since all rules are idempotent and the specification ST' does not contain any recursive function.

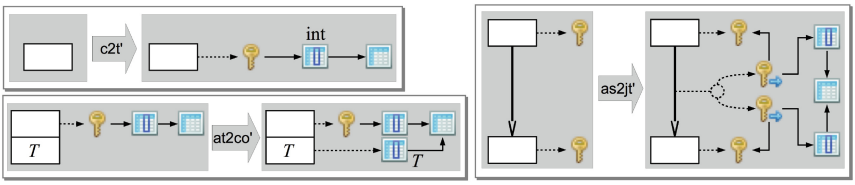

Fig. 8. Epic handling of classes, attributes, and associations in ST (and CT)

[35] The notation indicates that we assume the same carrier for Basetype in CM and RS.

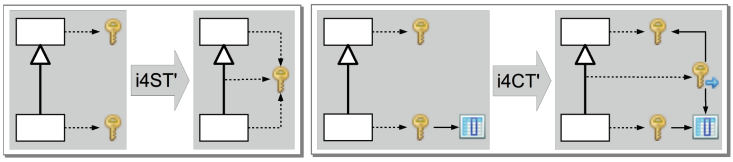

Fig. 9. Epic handling of inheritance in ST and CT

From the practical point of view, `ST'` allows more compact rules (compare Figs. 2 and 9.) and is a more precise description for the mapping of class model items to elements in a relational schema than `ST`: Classes are mapped to keys, since the only feature classes provide for their objects is *object identity*. Attributes are mapped to columns as before. And associations are mapped to foreign key pairs, since it is the key pair that enforces *type conformance* of the association's links in the relational model.

From the theoretical point of view, `ST'` and the corrected, now epic rules are better than the rules in Sect. 2, since they induce an epi-reflection, are idempotent, and, therefore, can easily be analysed wrt. termination.

The two other patterns `CT` and `CCT`, discussed in Sect. 2, can also be turned into epi-reflections. The pattern `CT` differs from `ST` just by the handling of inheritance. The corresponding transformation rule is `i4CT'` in Fig. 9. The necessary mapping of inheritance relations to relational schemata can be provided by a partial operation `I2FK : Inheritance ⟶ ForeignKey`.

Specification 4. CCT as parametric specification

```
Source := CM;   Target := Source +Basetype RS +
opns          C2K:              Class ⟶ Key
           At2Co:   Attribute,Class ⟶ Column
            As2T:        Association ⟶ Table          [x:As::==> As2T(x)]
     AS2FK,AT2FK: Association,Class ⟶ ForeignKey
axms           x:C::conc(c) ==> t(c(C2K(x))) = int                    (a5)
     x:At;y:C;z:T::z = ta(c(C2K(y))), <=(y,o(x)) ==>
                              ta(At2Co(x,y)) = z, t(At2Co(x,y)) = t(x)  (a6)
     x:As,y:C,z:K::<=(y,o(x)), z = C2K(y) ==> r(AS2FK(x,y)) = z,
                        n(c(AS2FK(x,y))), ta(c(AS2FK(x,y))) = As2T(x)  (a7)
     x:As,y:C,z:K::<=(y,t(x)), z = C2K(y) ==> r(AT2FK(x,y)) = z,
                        n(c(AT2FK(x,y))), ta(c(AT2FK(x,y))) = As2T(x)  (a8)
```

Specification 4 presents `CCT` as a parametric specification in the sense of [10].[36] The five rules for the transformation of class models into relation schemata are specified by the total operation `As2T` and the axioms `(a5)` – `(a8)`. The semantics of such a specification is the free construction from `Sys(Source)` to

[36] Again, `Source` and `Target` share sort `Basetype` with defined constant `int`.

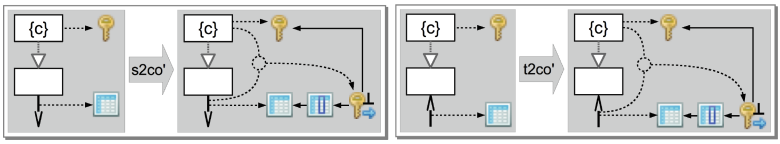

Fig. 10. Epic generation of foreign keys for associations in CCT

Sys(`Target`) wrt. the obvious forgetful functor in the opposite direction. The point-wise construction of these free objects is described in Sect. 3.2.

It is obvious that the presentation of constructive axioms as Horn-formulae is not as suggestive as the presentation as visual transformation rules, compare for example Fig. 10 which graphically depicts the axioms `(a7)` and `(a8)` by rules `s2co'` and `t2co'` respectively. But as we have shown in this paper both variants are semantically equivalent, if the presented morphisms are epic.

6 Summary

In this paper, we discussed the close connection between generating graph rules and the point-wise construction of epi-reflectors for finite constructive axioms. Constructive axioms turned out to be special cases of arbitrary generating rules. And a finite transformation from an arbitrary object o to a final one can be interpreted as the calculation of the epi-reflector of o, if all rules are epimorphisms.

In categories of partial algebras, constructive axioms can (operation-) generate new elements (e.g. rules in Fig. 10), can add new predicate definitions (e.g. rules in Fig. 4) and are able to identify items (e.g. rule `i4ST'` in Fig. 9). Therefore, constructive axioms in categories of partial algebras are suitable for the application area of model transformations for two reasons.

First of all, model transformation rules are typically generating rules, compare for example Triple Graph Grammars (TGG) [2,9], which have been proposed as a standard framework for model transformation. All rules in TGG are generating, especially all sets of derived rules that can be used for model transformation, i.e. forward, backward, source/target, and integration rules. Future research will investigate the connection between TGG and the framework proposed here.

Secondly, a model transformation produces *the derived* target model for a given source model, i.e. the target shall be *uniquely determined* (possibly up to isomorphism) for each source model and it shall be computable in a *finite* number of steps. If the computation is a calculation of an epi-reflector, uniqueness is for free. And, as we showed above, constructive axioms show better termination behaviour than arbitrary rules. We demonstrated these features in this paper by some typical examples. Future research, especially the elaboration of more and bigger transformation examples, will show, if epimorphisms are sufficient for model transformation.

References

1. Adámek, J., Herrlich, H., Strecker, G.E.: Abstract and Concrete Categories - The Joy of Cats (2004). http://katmat.math.uni-bremen.de/acc
2. Anjorin, A., Leblebici, E., Schürr, A.: 20 years of triple graph grammars: a roadmap for future research. ECEASST, 73 (2015)
3. Burmeister, P.: Introduction to theory and application of partial algebras - Part I. Mathematical Research, vol. 32. Akademie-Verlag, Berlin (1986)
4. Chen, P.P.: The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. **1**(1), 9–36 (1976)
5. Claßen, I., Ehrig, H., Wolz, D.: Algebraic specification techniques and tools for software development: the act approach. AMAST Series in Computing, vol. 1. World Scientific, River Edge (1993)
6. World Wide Web Consortium. Xml Schema, W3C (2012). https://www.w3.org/standards/techs/xmlschema
7. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT 2006. LNCS, vol. 4178, pp. 30–45. Springer, Heidelberg (2006). https://doi.org/10.1007/11841883_4
8. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Springer, Heidelberg (2006)
9. Ehrig, H., Ermel, C., Golas, U., Hermann, F.: Graph and Model Transformation - General Framework and Applications. Springer, Monographs in Theoretical Computer Science. An EATCS Series (2015)
10. Ehrig, H., Mahr, B.: Fundamentals of algebraic specification 1: equations and initial semantics. EATCS Monographs on Theoretical Computer Science., vol. 6. Springer, Heidelberg (1985)
11. Ehrig, H., Mahr, B.: Fundamentals of algebraic specification 2. EATCS Monographs on Theoretical Computer Science, vol. 21. Springer, Heidelberg (1990)
12. Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: an algebraic approach. In: FOCS, pp. 167–180. IEEE (1973)
13. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley, Boston (2003)
14. Object Management Group: Business Process Model and Notation 2.0.2. OMG (2013). http://www.omg.org/spec/BPMN/index.htm
15. Object Management Group: Unified Modeling Language 2.5. OMG (2015). http://www.omg.org/spec/UML/
16. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Fundam. Inform. **26**(3/4), 287–313 (1996)
17. Harel, D.: Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987)
18. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theor. Comput. Sci. **109**(1&2), 181–224 (1993)
19. Löwe, M., Schulz, C.: Algebraic Systems. May 2017. http://ux-02.ha.bib.de/daten/Löwe/Master/TheorieInformationssystem/Algebra20170523.pdf
20. Petri, C.A., Reisig, W.: Petri net. Scholarpedia **3**(4), 6477 (2008)
21. Tempelmeier, M., Löwe, M.: Single-Pushout Transformation partieller Algebren. Technical Report 2015/1, FHDW-Hannover (2015) (in German)