On Normal Forms for Structured Specifications with Generating Constraints

On normal forms for structured specifications with generating constraints*

Donald Sannella[1] and Andrzej Tarlecki[2]

[1] Laboratory for Foundations of Computer Science, University of Edinburgh
[2] Institute of Informatics, University of Warsaw

Abstract. Hartmut Ehrig and others in [EWT83] studied normal form results for complex generating constraints imposed on basic specifications. Since then this work has been followed by subsequent results concerning normal forms for structured specifications, typically built from basic specifications using union, translation and hiding. We consider generating constraints as additional specification-building operations and follow and extend the results concerning normal forms for the resulting specifications with various forms of generating constraints.

1 Introduction

Hartmut Ehrig and others in [EWT83] studied normal form results for complex generating constraints imposed on basic specifications. Although from today's point of view the results were somewhat restricted in their generality, they spurred a line of work on normal forms of structured specifications, notably in [BHK90] and in the current general version in [Bor02], which turned out to be crucial in the study of proof systems for consequences of structured specifications, and in the analysis of completeness properties of such proof systems. But the more recent normal form results largely disregarded generating (or reachability) properties as imposed by the constraints studied in [EWT83]. Our aim here is to fill this gap, by generalising the results of [EWT83] as follows.

First, as has been standard since the introduction of institutions [GB84,GB92] to free algebraic specifications from dependency on a specific logical system, we abstract away from the specifics of the underlying logical system and present our results in the framework of a rather arbitrary logical system formalised as an institution with minimal extra structure and assumed properties.

Then, we consider a normal form of specifications that is somewhat more restrictive than the canonical constraints in [EWT83], giving a normal form result that is a bit sharper than the corresponding result in [EWT83].

Finally, and most crucially, Ehrig *et al.* [EWT83] studied generating constraints that impose generation requirements within models of a "flat" basic specification (presentation) only. We study generating constraints imposed by a more general specification building operation, which may be mixed with other

* This work has been partially supported by the (Polish) National Science Centre, grant 2013/11/B/ST6/01381 (AT).

specification-building operations in an arbitrary way, so that generating requirements may be imposed in multiple "layers" within models of an arbitrarily complex specification. This makes the study more delicate, and in fact the normal form result one might expect does not carry over to this more general case.

Dedication: *This study is dedicated to the memory of Hartmut Ehrig. We are grateful to Hartmut for his kindness and generosity over the years. We represented different "schools" of thought on algebraic specification, but Hartmut was always friendly and willing to explain his ideas and to try to understand our point of view.*

2 Constraints in the standard algebraic framework

We begin with a summary of [EWT83].

As was usual at the time, the investigation was carried out in the context of the standard algebraic framework [EM85]. Specifications are *presentations* $\langle \Sigma, \Phi \rangle$ where Σ is a standard many-sorted signature and Φ is a set of Σ-axioms, usually equations. The category of Σ-algebras $\mathbf{Alg}(\Sigma)$ is defined as usual, with the notion of satisfaction between algebras and axioms yielding the obvious semantics of presentations: $[\![\langle \Sigma, \Phi \rangle]\!] = \{A \in |\mathbf{Alg}(\Sigma)| \mid A \models \Phi\}$.

The category of algebraic signatures **AlgSig** with standard signature morphisms $\sigma\colon \Sigma \to \Sigma'$ is cocomplete. For each signature morphism $\sigma\colon \Sigma \to \Sigma'$, we have a σ-reduct functor $U_\sigma\colon \mathbf{Alg}(\Sigma') \to \mathbf{Alg}(\Sigma)$. A signature morphism $\sigma\colon \Sigma \to \Sigma'$ is a *presentation morphism* $\sigma\colon \langle \Sigma, \Phi \rangle \to \langle \Sigma', \Phi' \rangle$ if $U_\sigma([\![\langle \Sigma', \Phi' \rangle]\!]) \subseteq [\![\langle \Sigma, \Phi \rangle]\!]$. Colimits lift from the category of signatures to the category of presentations. The crucial satisfaction condition and amalgamation properties hold as expected — see Sect. 3 for more general formulations, or check for instance [ST12].

For each presentation $PRES$, the authors of [EWT83] introduce *constraints* on $PRES$, which are built from the empty constraint $\emptyset$, with semantics $[\![\emptyset]\!] = [\![PRES]\!]$, using *union*, with $[\![C_1 + C_2]\!] = [\![C_1]\!] \cap [\![C_2]\!]$, and the following constructors, for any presentation morphisms $\sigma\colon PRES' \to PRES$, $\sigma'\colon PRES \to PRES'$ and constraint C' on $PRES'$:

- translation TRA_σ: $[\![\mathrm{TRA}_\sigma(C')]\!] = \{A \in [\![PRES]\!] \mid U_\sigma(A) \in [\![C']\!]\}$
- reflection $\mathrm{REF}_{\sigma'}$: $[\![\mathrm{REF}_{\sigma'}(C')]\!] = \{U_{\sigma'}(A') \mid A' \in [\![C']\!]\}$
- generating constraint GEN_σ: $[\![\mathrm{GEN}_\sigma(C')]\!] = \{A \in [\![PRES]\!] \mid U_\sigma(A) \in [\![C']\!]$ and A is U_σ-generated in $[\![PRES]\!]\}$, where $A \in [\![PRES]\!]$ is *U_σ-generated in* $[\![PRES]\!]$ if in $[\![PRES]\!]$ there is no proper subalgebra of A with the same σ-reduct as A.

A constraint on $PRES$ of the form $\mathrm{REF}_{\sigma_3}(\mathrm{TRA}_{\sigma_2}(\mathrm{GEN}_{\sigma_1}(\emptyset)))$ is called *canonical*. Such constraints might be easier to read in the diagrammatic notation of [EWT83]:

$$PRES_1 \xrightarrow[\mathrm{GEN}]{\sigma_1} PRES_2 \xrightarrow[\mathrm{TRA}]{\sigma_2} PRES_3 \xleftarrow[\mathrm{REF}]{\sigma_3} PRES$$

for presentations $PRES_i$ and presentation morphisms σ_i, $i = 1, 2, 3$.

The key result in [EWT83] is the following normal form theorem:

Theorem 2.1 ([EWT83]). *For each constraint on PRES an equivalent canonical constraint may be constructed.* □

We found the result very interesting and analysed it in detail already at the time [Tar83]. One observation was that the above form of canonical constraints is the only one possible for Thm. 2.1 to hold — that is, no other order of generating constraints, translation and reflection would work. Another was that some of the assumptions in [EWT83] are either misleading or unnecessary. For instance, the authors require reduct functors to lift isomorphisms, which in the standard algebraic framework excludes presentation morphisms that are not injective on sorts. Even if we could accept this as a reasonable restriction, it turns out that this property cannot be maintained under the constructions used in the proof of Thm. 2.1. Fortunately, such details did not prove to be crucial for the correctness of the proof, and the paper and the above theorem influenced subsequent developments, most notably the simpler normal form results for specifications without generating constraints in [BHK90] and [Bor02] and much work based in turn on those results.

3 Institutions with model inclusions

To capture in a very general way the concept of a submodel, we will require our model categories to come with *inclusions*.

A class of morphisms in a category is called a class of *inclusions* if it imposes a partial order on the objects of the category; to be precise, we require that

- all identities are inclusions
- inclusions are closed under composition
- between any two objects there is at most one inclusion (in either direction, i.e., for objects A, B, either there is no inclusion between A and B, or there is a unique inclusion from A to B, or from B to A, but not both unless A and B coincide).

In other words, a *category with inclusions* is a pair $\mathbb{C} = \langle \mathcal{C}, \mathcal{I} \rangle$ such that $\mathcal{C}$ is a category and $\mathcal{I}$ is a wide thin skeletal subcategory of $\mathcal{C}$; the morphisms of $\mathcal{I}$ are called *inclusions*. If there is an inclusion $\iota \colon A \to B$ then we say that A is a *subobject* of B and write $A \subseteq B$.

When no confusion may arise, we use the standard categorical terminology and notation in $\mathbb{C}$ to refer to the corresponding concepts in $\mathcal{C}$. So, for instance, we write $|\mathbb{C}|$ for the class $|\mathcal{C}|$ of objects in $\mathcal{C}$, by a diagram in $\mathbb{C}$ we mean a diagram in $\mathcal{C}$, by (co)limits in $\mathbb{C}$ we mean (co)limits in $\mathcal{C}$, etc.

A functor between categories with inclusions $F \colon \langle \mathcal{C}, \mathcal{I} \rangle \to \langle \mathcal{C}', \mathcal{I}' \rangle$ is a functor $F \colon \mathcal{C} \to \mathcal{C}'$ that preserves inclusions, $F(\mathcal{I}) \subseteq \mathcal{I}'$. Clearly, such functors compose, and so this yields the (quasi-)category **ICat** of categories with inclusions.

These are extremely mild requirements concerning the subcategory of inclusions. For instance, in contrast to some other work, e.g. [DGS93], [CR97], [GR04] and [CMST17], we have no need to assume that inclusions form part of a (strict) factorisation system for $\mathcal{C}$, or that they admit intersections and unions, etc.

An *institution with model inclusions* **INS** consists of:

- a category $\mathbf{Sign_{INS}}$ of *signatures*;
- a functor $\mathbf{Sen_{INS}}\colon \mathbf{Sign_{INS}} \to \mathbf{Set}$, giving a set $\mathbf{Sen_{INS}}(\Sigma)$ of *Σ-sentences* for each signature $\Sigma \in |\mathbf{Sign_{INS}}|$; and a function $\mathbf{Sen_{INS}}(\sigma)\colon \mathbf{Sen_{INS}}(\Sigma) \to \mathbf{Sen_{INS}}(\Sigma')$ which translates Σ-sentences to Σ'-sentences for each signature morphism $\sigma\colon \Sigma \to \Sigma'$;
- a functor $\mathbf{Mod_{INS}}\colon \mathbf{Sign}^{op}_{\mathbf{INS}} \to \mathbf{ICat}$, giving a category $\mathbf{Mod_{INS}}(\Sigma)$ of *Σ-models with model inclusions* for each signature $\Sigma \in |\mathbf{Sign_{INS}}|$; and a functor $\mathbf{Mod_{INS}}(\sigma)\colon \mathbf{Mod_{INS}}(\Sigma') \to \mathbf{Mod_{INS}}(\Sigma)$ which translates Σ'-models to Σ-models and Σ'-morphisms to Σ-morphisms, preserving model inclusions, for each signature morphism $\sigma\colon \Sigma \to \Sigma'$; and
- a family $\langle \models_{\mathbf{INS},\Sigma} \subseteq |\mathbf{Mod_{INS}}(\Sigma)| \times \mathbf{Sen_{INS}}(\Sigma)\rangle_{\Sigma \in |\mathbf{Sign_{INS}}|}$ of *satisfaction relations*

such that for any signature morphism $\sigma\colon \Sigma \to \Sigma'$ the translations $\mathbf{Mod_{INS}}(\sigma)$ of models and $\mathbf{Sen_{INS}}(\sigma)$ of sentences preserve the satisfaction relation, that is, for any $\varphi \in \mathbf{Sen_{INS}}(\Sigma)$ and $M' \in |\mathbf{Mod_{INS}}(\Sigma')|$ the following *satisfaction condition* holds:

$$M' \models_{\mathbf{INS},\Sigma'} \mathbf{Sen_{INS}}(\sigma)(\varphi) \quad \text{iff} \quad \mathbf{Mod_{INS}}(\sigma)(M') \models_{\mathbf{INS},\Sigma} \varphi$$

Note that institutions with model inclusions are not "inclusive institutions" in the sense of [DGS93], [GR04] and [CMST17]: we require inclusion structure on models, not on signatures.

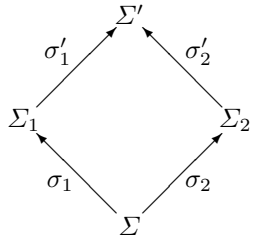
Examples of institutions with model inclusions abound. The institution **EQ** of equational logic has many-sorted algebraic signatures as signatures, many-sorted algebras as models with the usual notion of subalgebra determining the model inclusions and (explicitly quantified) equations as sentences. The institution **FOPEQ** of first-order predicate logic with equality has signatures that add predicate names to many-sorted algebraic signatures, models that extend algebras by interpreting predicate names as relations with inclusions that are required to preserve these relations, and sentences that are all closed (no free variables) formulae of first-order logic with equality. See [ST12] for detailed definitions of these and many other institutions, which often can be enriched with the obvious concept of a submodel, to yield institutions with model inclusions.

We will freely use standard terminology, and say that a Σ-model M *satisfies* a Σ-sentence φ, or that φ *holds* in M, whenever $M \models_{\mathbf{INS},\Sigma} \varphi$. We will omit the subscript **INS**, writing $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$. Similarly, the subscript Σ on the satisfaction relations will often be omitted. For any signature morphism $\sigma\colon \Sigma \to \Sigma'$, the translation function $\mathbf{Sen}(\sigma)\colon \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ will be denoted by $\sigma\colon \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$, the coimage function w.r.t. $\mathbf{Sen}(\sigma)$ by $\sigma^{-1}\colon \mathcal{P}(\mathbf{Sen}(\Sigma')) \to \mathcal{P}(\mathbf{Sen}(\Sigma))$, and the reduct functor $\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$ by $\_|_\sigma\colon \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$. Thus, the satisfaction condition may be re-stated as: $M' \models \sigma(\varphi)$ iff $M'|_\sigma \models \varphi$.

An institution with inclusions **INS** is *(finitely) exact* if its category **Sign** of signatures is (finitely) cocomplete, and the functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \to \mathbf{ICat}$ maps

(finite) colimits of signatures to limits of model categories with inclusions in **ICat**. This adjusts the usual notion of exactness [ST12] to the framework where model inclusions are considered. In particular, the following stronger form of the amalgamation property [EM85] holds:

Lemma 3.1. *Given a finitely exact institution with model inclusions* **INS***, consider a pushout of signatures*



Then, for any $M_1 \in |\mathbf{Mod}(\Sigma_1)|$ *and* $M_2 \in |\mathbf{Mod}(\Sigma_2)|$ *such that* $M_1|_{\sigma_1} = M_2|_{\sigma_2}$*, there exists a unique* $M' \in |\mathbf{Mod}(\Sigma')|$ *such that* $M'|_{\sigma'_1} = M_1$ *and* $M'|_{\sigma'_2} = M_2$*. Moreover, for any submodels* $N_1 \subseteq M_1$ *and* $N_2 \subseteq M_2$ *such that* $N_1|_{\sigma_1} = N_2|_{\sigma_2}$ *(hence* $N_1|_{\sigma_1} = N_2|_{\sigma_2} \subseteq M_1|_{\sigma_1} = M_2|_{\sigma_2}$*) the unique* $N' \in |\mathbf{Mod}(\Sigma')|$ *such that* $N'|_{\sigma'_1} = N_1$ *and* $N'|_{\sigma'_2} = N_2$ *is a submodel of* M'*,* $N' \subseteq M'$*.* □

The standard institutions with model inclusions mentioned above (**EQ**, **FOPEQ**, etc.) are exact, hence enjoy the amalgamation property captured by Lemma 3.1.

Given a signature morphism $\sigma\colon \Sigma \to \Sigma'$ and class $\mathcal{M}' \subseteq |\mathbf{Mod}(\Sigma')|$ of models, we say that a model $M' \in |\mathbf{Mod}(\Sigma')|$ is *σ-generated in $\mathcal{M}'$* if $M' \in \mathcal{M}'$ and it has no proper submodels in $\mathcal{M}'$ with the same σ-reduct: for any submodel $M'' \subseteq M'$ if $M'' \in \mathcal{M}'$ and $M''|_\sigma = M'|_\sigma$ then $M'' = M'$. By taking $\mathcal{M}' = |\mathbf{Alg}(\Sigma')|$ we obtain the standard definition of σ-generated model, which requires M' to be generated by the set of all its elements in the carriers of $M'|_\sigma$. But note that when $\mathcal{M}' \subsetneq |\mathbf{Alg}(\Sigma')|$ — in particular, when $\mathcal{M}'$ is not closed under submodels — models that are generated in $\mathcal{M}'$ need not be generated in the standard sense, exactly as in [EWT83]. For this reason a better terminology might be "minimal", as in [ST88], but we retain the terminology of [EWT83] to avoid confusion.

4 Structured specifications in institutions with model inclusions

Taking an institution as a starting point for talking about specifications, each signature Σ captures static information about the interface of a software system with each Σ-model representing a possible realisation of such a system, and with Σ-sentences used to describe properties that a realisation is required to satisfy. As a consequence, it is natural to regard the meaning of any specification SP built in an institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ as given by its signature $Sig[SP] \in |\mathbf{Sign}|$ together with a class $Mod[SP]$ of $Sig[SP]$-models. Specifications SP with $Sig[SP] = \Sigma$ are referred to as *Σ-specifications*.

The semantics of specifications yields the obvious notion of specification equivalence: two specifications SP_1 and SP_2 are *equivalent*, written $SP_1 \equiv SP_2$, if $Sig[SP_1] = Sig[SP_2]$ and $Mod[SP_1] = Mod[SP_2]$.

Specifications we will consider are built from basic specifications (presentations in **INS**) using *specification-building operations* [ST12]. Specification formalisms differ in the choice of these operations, but typically all share a kernel introduced in ASL [SW83,ST88], where specifications are built from *basic specifications* using *union*, *translation*, and *hiding*. Following [EWT83], we add *generating contraints* to this repertoire, to capture constraints as studied there via a more general specification-building operation. We use a syntax inspired by that of CASL [BM04].

***basic specifications*:** For any signature $\Sigma \in |\mathbf{Sign}|$ and set $\Phi \subseteq \mathbf{Sen}(\Sigma)$ of Σ-sentences, a *basic specification* $\langle \Sigma, \Phi \rangle$ is a specification with:

$Sig[\langle \Sigma, \Phi \rangle] = \Sigma$

$Mod[\langle \Sigma, \Phi \rangle] = \{M \in \mathbf{Mod}(\Sigma) \mid M \models \Phi\}$

***union*:** For any signature $\Sigma \in |\mathbf{Sign}|$, given Σ-specifications SP_1 and SP_2, their *union* $SP_1 \cup SP_2$ is a specification with:

$Sig[SP_1 \cup SP_2] = \Sigma$

$Mod[SP_1 \cup SP_2] = Mod[SP_1] \cap Mod[SP_2]$

***translation*:** For any signature morphism $\sigma\colon \Sigma \to \Sigma'$ and Σ-specification SP, SP **with** σ is a specification with:

$Sig[SP \textbf{ with } \sigma] = \Sigma'$

$Mod[SP \textbf{ with } \sigma] = \{M' \in |\mathbf{Mod}(\Sigma')| \mid M'|_\sigma \in Mod[SP]\}$

***hiding*:** For any signature morphism $\sigma\colon \Sigma \to \Sigma'$ and Σ'-specification SP', SP' **hide via** σ is a specification with:

$Sig[SP' \textbf{ hide via } \sigma] = \Sigma$

$Mod[SP' \textbf{ hide via } \sigma] = \{M'|_\sigma \mid M' \in Mod[SP']\}$

***generating constraints*:** For any signature morphism $\sigma\colon \Sigma \to \Sigma'$, Σ-specification SP and Σ'-specification SP', **generate by** σ **from** SP **in** SP' is a specification with:

$Sig[\textbf{generate by } \sigma \textbf{ from } SP \textbf{ in } SP'] = \Sigma'$

$Mod[\textbf{generate by } \sigma \textbf{ from } SP \textbf{ in } SP'] =$
$\{M' \in |\mathbf{Mod}(\Sigma')| \mid M' \text{ is } \sigma\text{-generated in } Mod[SP'], M'|_\sigma \in Mod[SP]\}$

The above specification-building operations may be arbitrarily combined to derive additional specification-building operations that capture some common patterns of their use. For instance:

***enrichment*:** For any specification SP and a set $\Phi \subseteq \mathbf{Sen}(Sig[SP])$ of sentences, SP **then** Φ abbreviates $SP \cup \langle Sig[SP], \Phi \rangle$.

5 Algebraic properties of specification-building operations

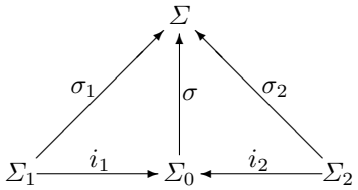
We start by recalling some easy facts concerning the standard specification-building operations [ST12]:

Proposition 5.1. *In any institution, under the obvious requirements on specification signatures, sentences and signature morphisms involved to ensure well-formedness of the specifications concerned:*

1. $\langle \Sigma, \Phi_1 \rangle \cup \langle \Sigma, \Phi_2 \rangle \equiv \langle \Sigma, \Phi_1 \cup \Phi_2 \rangle$
2. $\langle \Sigma, \Phi \rangle$ **with** $\sigma \colon \Sigma \to \Sigma' \equiv \langle \Sigma', \sigma(\Phi) \rangle$
3. SP **with** $id_{Sig[SP]} \equiv SP \equiv SP$ **hide via** $id_{Sig[SP]}$
4. $(SP$ **with** $\sigma)$ **with** $\sigma' \equiv SP$ **with** $\sigma;\sigma'$
5. $(SP$ **hide via** $\sigma)$ **hide via** $\sigma' \equiv SP$ **hide via** $\sigma';\sigma$
6. $(SP \cup SP')$ **with** $\sigma \equiv (SP$ **with** $\sigma) \cup (SP'$ **with** $\sigma)$ □

The following easy fact follows directly from Prop. 5.1(4 and 6):

Proposition 5.2. *In any institution* **INS**, *consider the following commuting diagram of signatures:*

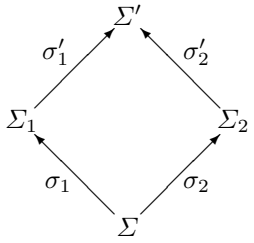


Then for any Σ_1-specification SP_1 and Σ_2-specification SP_2,

$$(SP_1 \textbf{ with } \sigma_1) \cup (SP_2 \textbf{ with } \sigma_2) \equiv ((SP_1 \textbf{ with } i_1) \cup (SP_2 \textbf{ with } i_2)) \textbf{ with } \sigma$$

□

Proposition 5.3. *In any finitely exact institution* **INS**, *given a pushout of signatures*



1. $(SP_1$ **hide via** $\sigma_1)$ **with** $\sigma_2 \equiv (SP_1$ **with** $\sigma'_1)$ **hide via** σ'_2, *for any Σ_1-specification SP_1, and*
2. $(SP_1$ **hide via** $\sigma_1) \cup (SP_2$ **hide via** $\sigma_2) \equiv$
$((SP_1$ **with** $\sigma'_1) \cup (SP_2$ **with** $\sigma'_2))$ **hide via** $\sigma_1;\sigma'_1$,
for any Σ_1-specification SP_1 and Σ_2-specification SP_2.

Proof. See Props. 5.6.5 and 5.6.7 in [ST12]. □

We can also derive algebraic properties for derived specification-building operations; for instance the following property follows directly from Prop. 5.1(6 and 2):

Proposition 5.4. *For any specification SP, set $\Phi \subseteq \mathbf{Sen}(Sig[SP])$ of sentences and signature morphism $\sigma\colon Sig[SP] \to \Sigma'$,*

$$(SP \textbf{ then } \Phi) \textbf{ with } \sigma \equiv (SP \textbf{ with } \sigma) \textbf{ then } \sigma(\Phi) \qquad \square$$

In **generate by** σ **from** SP **in** SP', both the "source" specification SP and the "target" specification SP' may be arbitrarily complex, built using any combination of specification-building operations, including generating constraints. This is considerably more general than the constraints considered in [EWT83], where in particular the "target" specification, within which we select the generated models, was taken to be a basic specification.

To begin with, let us note that the complexity of the source specification in a generating constraint may easily be removed. The source specification does not affect the generation property of the models within the target specification, and so it may be moved out of the scope of the constraint and imposed separately:

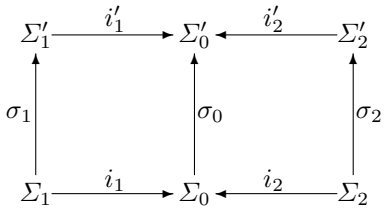
Proposition 5.5. *In any institution with model inclusions* **INS***, for any signature morphism $\sigma\colon \Sigma \to \Sigma'$, Σ-specification SP and Σ'-specification SP',*

$$\begin{aligned}&\textbf{generate by } \sigma \textbf{ from } SP \textbf{ in } SP' \equiv \\ &\qquad (\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } SP') \cup (SP \textbf{ with } \sigma)\end{aligned}$$

Proof. From the definition of the semantics of the specification-building operations involved. $\square$

The following property will be used to combine generating constraints:

Lemma 5.6. *Consider two constraints* **generate by** σ_1 **from** SP_1 **in** SP'_1 *and* **generate by** σ_2 **from** SP_2 **in** SP'_2*, where $Sig[SP_1] = \Sigma_1$, $Sig[SP'_1] = \Sigma'_1$, $Sig[SP_2] = \Sigma_2$, $Sig[SP'_2] = \Sigma'_2$. Let Σ_0 be a coproduct of Σ_1 and Σ_2 with injections $i_1\colon \Sigma_1 \to \Sigma_0$ and $i_2\colon \Sigma_2 \to \Sigma_0$, and Σ'_0 be a coproduct of Σ'_1 and Σ'_2 with injections $i'_1\colon \Sigma'_1 \to \Sigma'_0$ and $i'_2\colon \Sigma'_2 \to \Sigma'_0$, and let $\sigma_0\colon \Sigma_0 \to \Sigma'_0$ be the unique signature morphism such that $i_1;\sigma_0 = \sigma_1;i'_1$ and $i_2;\sigma_0 = \sigma_2;i'_2$.*



*Then for any model $M' \in |\mathbf{Mod}(\Sigma'_0)|$, M' is σ_0-generated in $Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$ iff both $M'|_{i'_1}$ is σ_1-generated in $Mod[SP'_1]$ and $M'|_{i'_2}$ is σ_2-generated in $Mod[SP'_2]$.*

*Proof. For the "only if" part, consider a model $M' \in |\mathbf{Mod}(\Sigma'_0)|$ such that M' is σ_0-generated in $Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$. Consider a submodel $N'_1 \subseteq M'|_{i'_1}$ such that $N'_1 \in Mod[SP'_1]$ and $N'_1|_{\sigma_1} = (M'|_{i'_1})|_{\sigma_1}$. Let $N' \in |\mathbf{Mod}(\Sigma'_0)|$ be (the unique model) such that $N'|_{i'_1} = N'_1$ and $N'|_{i'_2} = M'|_{i'_2}$. Then*

$N' \subseteq M'$ *(by Lemma 3.1),* $N' \in Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$ *(from the definition of the semantics of structured specifications), and* $N'|_{\sigma_0} = M'|_{\sigma_0}$ *(since* $(N'|_{\sigma_0})|_{i_2} = (N'|_{i'_2})|_{\sigma_2} = (M'|_{i'_2})|_{\sigma_2} = (M'|_{\sigma_0})|_{i_2}$*, and* $(N'|_{\sigma_0})|_{i_1} = (N'|_{i'_1})|_{\sigma_1} = (M'|_{i'_1})|_{\sigma_1} = (M'|_{\sigma_0})|_{i_1}$*). Hence* $N' = M'$*, and so* $N'_1 = M'|_{i'_1}$*, which shows that* $M'|_{i'_1}$ *is* σ_1*-generated in* $Mod[SP'_1]$*. By symmetry,* $M'|_{i'_2}$ *is* σ_2*-generated in* $Mod[SP'_2]$*.*

For the opposite implication, suppose both $M'|_{i'_1}$ *is* σ_1*-generated in* $Mod[SP'_1]$ *and* $M'|_{i'_2}$ *is* σ_2*-generated in* $Mod[SP'_2]$*. Consider a submodel* $N' \subseteq M'$ *such that* $N' \in Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$ *and* $N'|_{\sigma_0} = M'|_{\sigma_0}$*. Then* $N'|_{i'_1} \in Mod[SP'_1]$ *is a submodel of* $M'|_{i'_1}$ *such that* $(N'|_{i'_1})|_{\sigma_1} = (N'|_{\sigma_0})|_{i_1} = (M'|_{\sigma_0})|_{i_1} = (M'|_{i'_1})|_{\sigma_1}$*. Therefore* $N'|_{i'_1} = M'|_{i'_1}$*. By symmetry,* $N'|_{i'_2} = M'|_{i'_2}$*. Hence* $N' = M'$*, which shows* M' *is indeed* σ_0*-generated in* $Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$*.* □

Corollary 5.7. *Under the notation of Lemma 5.6:*

$$\left(\begin{array}{l}((\textbf{generate by } \sigma_1 \textbf{ from } SP_1 \textbf{ in } SP'_1) \textbf{ with } i'_1) \cup \\ \quad ((\textbf{generate by } \sigma_2 \textbf{ from } SP_2 \textbf{ in } SP'_2) \textbf{ with } i'_2)\end{array}\right) \equiv$$
$$\textbf{generate by } \sigma_0 \textbf{ from } \left(\begin{array}{r}(SP_1 \textbf{ with } i_1) \cup \\ (SP_2 \textbf{ with } i_2)\end{array}\right) \textbf{ in } \left(\begin{array}{r}(SP'_1 \textbf{ with } i'_1) \cup \\ (SP'_2 \textbf{ with } i'_2)\end{array}\right)$$

Proof. Let SP_l *be the specification on the left-hand side of the equivalence, and let* SP_r *be the specification on its right-hand side.*

Consider $M'_0 \in Mod[SP_l]$*. Then* $M'_0|_{i'_1}$ *is* σ_1*-generated in* $Mod[SP'_1]$ *and* $M'_0|_{i'_2}$ *is* σ_2*-generated in* $Mod[SP'_2]$*. Hence, by Lemma 5.6,* M'_0 *is* σ_0*-generated in* $Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$*. Moreover,* $(M'_0|_{i'_1})|_{\sigma_1} = (M'_0|_{\sigma_0})|_{i_1} \in Mod[SP_1]$ *and* $(M'_0|_{i'_2})|_{\sigma_2} = (M'_0|_{\sigma_0})|_{i_2} \in Mod[SP_2]$*, hence we have* $M'_0|_{\sigma_0} \in Mod[(SP_1$ **with** $i_1) \cup (SP_2$ **with** $i_2)]$*. Thus,* $M'_0 \in Mod[SP_r]$*.*

Consider now $M'_0 \in Mod[SP_r]$*.* M'_0 *is* σ_0*-generated in* $Mod[(SP'_1$ **with** $i'_1) \cup (SP'_2$ **with** $i'_2)]$*, hence by Lemma 5.6,* $M'_0|_{i'_1}$ *is* σ_1*-generated in* $Mod[SP'_1]$ *and* $M'_0|_{i'_2}$ *is* σ_2*-generated in* $Mod[SP'_2]$*. Moreover,* $M'_0|_{\sigma_0} \in Mod[(SP_1$ **with** $i_1) \cup (SP_2$ **with** $i_2)]$*, hence* $(M'_0|_{i'_1})|_{\sigma_1} = (M'_0|_{\sigma_0})|_{i_1} \in Mod[SP_1]$ *and* $(M'_0|_{i'_2})|_{\sigma_2} = (M'_0|_{\sigma_0})|_{i_2} \in Mod[SP_2]$*. Thus* $M'_0|_{i'_1} \in Mod[$**generate by** σ_1 **from** SP_1 **in** $SP'_1]$ *and* $M'_0|_{i'_2} \in Mod[$**generate by** σ_2 **from** SP_2 **in** $SP'_2]$*, which shows that* $M'_0 \in Mod[SP_l]$*.* □

6 Normal form results

We now come to the presentation of so-called normal-form results for structured specifications, whereby we show that all specifications of a certain kind are equivalent to a specification in a certain simple "normal form".

The first such result is easy:

Theorem 6.1. *In any institution* **INS***, for every specification SP built from basic specifications using union and translation, there is an equivalent (basic) specification of the form* $\langle \Sigma, \Phi \rangle$ *(where* Φ *is finite provided the basic specifications involved in SP are finite).*

Proof. By induction on the structure of specifications, using Prop. 5.1(1 and 2). □

Then, using Props. 5.1 and 5.3, one can show the following normal form result by an easy induction on the structure of specifications:

Theorem 6.2. *Let* **INS** *be a finitely exact institution. For every specification SP built from flat specifications using union, translation and hiding, there is an equivalent specification of the form* $\langle \Sigma, \Phi \rangle$ **hide via** σ *(where* Φ *is finite provided the basic specifications involved in SP are finite).*

Proof. See Theorem 5.6.10 in [ST12]. □

The above key result may be derived from the normal form result for constraints in [EWT83] (see Thm. 2.1 above). It was given in a very similar form in [BHK90] for the standard institution of first-order logic, and then generalised in [Bor02] to an arbitrary exact institution. The result proved crucial for a number of further foundational developments, notably in the study of completeness of standard logical systems for proving consequences of structured specifications.

However, unlike the normal form result in [EWT83], Thm. 6.2 does not address specifications with generating constraints. The key problem is to reduce the complexity of the specifications involved in the generating constraints.

A generating constraint **generate by** σ **from** SP **in** SP' is *source-trivial* if SP is a basic specification with no axioms, i.e., is of the form $\langle Sig[SP], \emptyset \rangle$. A constraint **generate by** σ **from** SP **in** SP' is *basic* if SP' is a basic specification.

As already mentioned, the complexity of the source specifications in generating constraints is not a problem:

Corollary 6.3. *In any institution* **INS** *with model inclusions, any structured specification built from basic specifications using union, translation, hiding and generating constraints is equivalent to a specification built from basic specifications using union, translation, hiding and source-trivial generating constraints.*

Proof. Follows from Prop. 5.5 by induction on the structure of specifications. □

We are now ready for a direct generalisation of Thm. 2.1, the main normal form result in [EWT83].

As recalled in Sect. 2, the canonical constraints of [EWT83] are of the form:

$$\langle \Sigma_1, \Phi_1 \rangle \xrightarrow[\text{GEN}]{\sigma_1} \langle \Sigma_2, \Phi_2 \rangle \xrightarrow[\text{TRA}]{\sigma_2} \langle \Sigma_3, \Phi_3 \rangle \xleftarrow[\text{REF}]{\sigma_3} \langle \Sigma, \Phi \rangle$$

In terms of the specification-building operations introduced in Sect. 4, this may be written as follows:

$$((((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \Phi_1 \rangle \textbf{ in } \langle \Sigma_2, \Phi_2 \rangle) \textbf{ with } \sigma_2) \textbf{ then } \Phi_3) \textbf{ hide via } \sigma_3) \textbf{ then } \Phi$$

with further requirements to ensure that the signature morphisms involved are in fact presentation morphisms. We will show below that the above form may be considerably simplified, by collecting all of the axioms involved in one place:

$$\langle \Sigma_1, \emptyset \rangle \xrightarrow[\text{GEN}]{\sigma_1} \langle \Sigma_2, \Phi_2 \rangle \xrightarrow[\text{TRA}]{\sigma_2} \langle \Sigma_3, \emptyset \rangle \xleftarrow[\text{REF}]{\sigma_3} \langle \Sigma, \emptyset \rangle$$

A specification is in *basic normal form* if it has the form:

$$((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma') \textbf{ hide via } \delta$$

where the signatures and signature morphisms are as in the following diagram:

$$\Sigma \xrightarrow{\sigma} \Sigma' \xrightarrow{\sigma'} \Sigma'' \xleftarrow{\delta} \widehat{\Sigma}$$

This makes Thm. 6.4 below stronger, even in the standard algebraic framework, than Thm. 2.1.

Theorem 6.4. *In any finitely exact institution* **INS** *with model inclusions, any structured specification built from basic specifications using union, translation, hiding and basic generating constraints is equivalent to a specification in basic normal form.*

Proof. First, by Cor. 6.3 (and Prop. 5.5) it is enough to consider structured specifications built from basic specifications using union, translation, hiding and source-trivial basic generating constraints. For those, we proceed by induction on the structure of specifications concerned, considering the last specification-building operation involved and assuming that its arguments, if any, are in basic normal form:

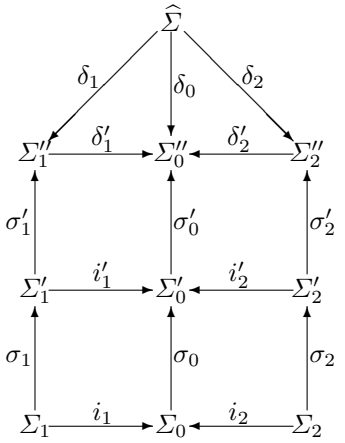
***basic specifications*:**

$\langle \Sigma, \Phi \rangle$

$\equiv$ *(directly by the semantics)*

$((\textbf{generate by } id_\Sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma, \Phi \rangle) \textbf{ with } id_\Sigma) \textbf{ hide via } id_\Sigma$

***union*:** *The following diagram in* **Sign** *may help to follow the equivalences below:*



$$
\begin{array}{l}
(((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma_1', \Phi_1' \rangle) \textbf{ with } \sigma_1') \textbf{ hide via } \delta_1) \\
\cup \\
(((\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } \langle \Sigma_2', \Phi_2' \rangle) \textbf{ with } \sigma_2') \textbf{ hide via } \delta_2) \\
\quad \equiv \textit{(by Prop. 5.3(2), taking a pushout } \Sigma_1'' \xrightarrow{\delta_1'} \Sigma_0'' \xleftarrow{\delta_2'} \Sigma_2'' \textit{ of the span} \\
\qquad\qquad \Sigma_1'' \xleftarrow{\delta_1} \widehat{\Sigma} \xrightarrow{\delta_2} \Sigma_2'' \textit{ and } \delta_0 = \delta_1;\delta_1' = \delta_2;\delta_2') \\
\left(\begin{array}{l} (((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma_1', \Phi_1' \rangle) \textbf{ with } \sigma_1') \textbf{ with } \delta_1') \\ \cup \\ (((\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } \langle \Sigma_2', \Phi_2' \rangle) \textbf{ with } \sigma_2') \textbf{ with } \delta_2') \end{array} \right) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{hide via } \delta_0 \\
\quad \equiv \textit{(by Prop. 5.1(4))} \\
\left(\begin{array}{l} ((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma_1', \Phi_1' \rangle) \textbf{ with } \sigma_1';\delta_1') \\ \cup \\ ((\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } \langle \Sigma_2', \Phi_2' \rangle) \textbf{ with } \sigma_2';\delta_2') \end{array} \right) \textbf{ hide via } \delta_0 \\
\quad \equiv \textit{(by Prop. 5.2, taking a coproduct } \Sigma_1' \xrightarrow{i_1'} \Sigma_0' \xleftarrow{i_2'} \Sigma_2' \textit{ and } \sigma_0' \colon \Sigma_0' \to \Sigma_0'' \\
\qquad\qquad \textit{such that } i_1';\sigma_0' = \sigma_1';\delta_1' \textit{ and } i_2';\sigma_0' = \sigma_2';\delta_2') \\
\left(\left(\begin{array}{l} ((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma_1', \Phi_1' \rangle) \textbf{ with } i_1') \\ \cup \\ ((\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } \langle \Sigma_2', \Phi_2' \rangle) \textbf{ with } i_2') \end{array} \right) \textbf{ with } \sigma_0' \right) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{hide via } \delta_0 \\
\quad \equiv \textit{(by Cor. 5.7, taking a coproduct } \Sigma_1 \xrightarrow{i_1} \Sigma_0 \xleftarrow{i_2} \Sigma_2 \textit{ and } \sigma_0 \colon \Sigma_0 \to \Sigma_0' \\
\qquad\qquad \textit{such that } i_1;\sigma_0 = \sigma_1;i_1' \textit{ and } i_2;\sigma_0 = \sigma_2;i_2', \textit{ and by Prop. 5.1(2 and 1))} \\
\big(\big((\textbf{generate by } \sigma_0 \textbf{ from } \langle \Sigma_0, \emptyset \rangle \textbf{ in } \langle \Sigma_0', i_1'(\Phi_1') \cup i_2'(\Phi_2') \rangle) \big) \big) \textbf{ with } \sigma_0' \big) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{hide via } \delta_0
\end{array}
$$

***translation*:**

$$
\begin{array}{l}
(((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma') \textbf{ hide via } \delta) \textbf{ with } \tau \\
\quad \equiv \textit{(by Prop. 5.3(1), taking a pushout } \Sigma' \xrightarrow{\tau'} \bullet \xleftarrow{\delta'} \widehat{\Sigma'} \textit{ of the span} \\
\qquad\qquad \Sigma' \xleftarrow{\delta} \widehat{\Sigma} \xrightarrow{\tau} \widehat{\Sigma'}) \\
(((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma') \textbf{ with } \tau') \textbf{ hide via } \delta' \\
\quad \equiv \textit{(by Prop. 5.1(4))} \\
((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma';\tau') \textbf{ hide via } \delta'
\end{array}
$$

***hiding*:**

$$
\begin{array}{l}
(((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma') \textbf{ hide via } \delta) \textbf{ hide via } \delta' \\
\quad \equiv \textit{(by Prop. 5.1(5))} \\
((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle) \textbf{ with } \sigma') \textbf{ hide via } \delta';\delta
\end{array}
$$

***source-trivial basic generating constraints*:**

$$
\begin{array}{l}
\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi' \rangle \\
\quad \equiv \textit{(by Prop. 5.1(3))} \\
((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \Phi \rangle) \textbf{ with } id_{\Sigma'}) \textbf{ hide via } id_{\Sigma'}
\end{array}
$$

□

One might expect that Thm. 6.4 can be generalised to cover arbitrary specifications built from basic specifications using union, translation, hiding and (not necessarily basic) generating constraints. Unfortunately, in general this need not be the case, as the following counterexample shows.

Counterexample 6.5. Consider a very simple institution $\mathbf{INS}_0$ with three signatures $\Sigma_\emptyset$, Σ_1 and Σ_2, and signature morphisms $\sigma\colon \Sigma_\emptyset \to \Sigma_1$, $\delta\colon \Sigma_1 \to \Sigma_2$ (and identities and the composition $\sigma;\delta$). This defines the signature category, which is finitely cocomplete. Let $\mathbf{Mod}_0(\Sigma_\emptyset)$ be a singleton. Then let $\mathbf{Mod}_0(\Sigma_1)$ contain three distinct models K_1, N_1 and M_1 such that $K_1 \subseteq N_1 \subseteq M_1$. Let $\mathbf{Mod}_0(\Sigma_2)$ have two distinct models N_2 and M_2 with no inclusions other than identities. We also put $N_2\big|_\delta = N_1$ and $M_2\big|_\delta = M_1$. Finally, let $\mathbf{Sen}(\Sigma_\emptyset) = \mathbf{Sen}_0(\Sigma_1) = \mathbf{Sen}_0(\Sigma_2) = \emptyset$.

Consider

$$\textbf{generate by } \sigma \textbf{ from } \langle \Sigma_0, \emptyset \rangle \textbf{ in } (\langle \Sigma_2, \emptyset \rangle \textbf{ hide via } \delta)$$

Clearly, N_1 is the only model of the above specification. By a simple analysis of all possible cases, no Σ-specification in basic normal form has N_1 as the only model: $\{N_1\}$ cannot be the model class of a specification without generating constraints, since N_1 and M_1 cannot be distinguished in such a specification. All basic generating constraints here admit all models of their target specifications, except for the following one:

$$\textbf{generate by } \sigma \textbf{ from } \langle \Sigma_\emptyset, \emptyset \rangle \textbf{ in } \langle \Sigma_1, \emptyset \rangle$$

which does not have N_1 as a model, since it has a proper submodel K_1. Hence, no specification in basic normal form built on this constraint has N_1 as a model.

Unfortunately, the above construction is not quite right: the institution we defined does not satisfy our assumptions, since the model functor is not continuous. For instance, the coproduct of Σ_1 and Σ_2 is Σ_2, but the class of its models is not a product of the model classes of Σ_1 and Σ_2. The overall idea of the counterexample works, but the following more complex construction is needed.

Consider a category $\mathcal{C}_0$ with two objects Σ_1 and Σ_2 and a morphism $\delta\colon \Sigma_1 \to \Sigma_2$, with model categories, reduct functor, and sets of sentences defined as above.

Let the category of signatures in $\mathbf{INS}'_0$ be the category $\mathbf{Set}^{\to}$ of morphisms in $\mathbf{Set}$, i.e. signatures now consist of two sets and a function between them, written $X_2 \xrightarrow{f} X_1$, and signature morphisms $\sigma\colon (X_2 \xrightarrow{f} X_1) \to (X'_2 \xrightarrow{f'} X'_1)$ are pairs of functions $\sigma_1\colon X_1 \to X'_1$ and $\sigma_2\colon X_2 \to X'_2$ such that $f;\sigma_1 = \sigma_2;f'$. It is well-known that $\mathbf{Set}^{\to}$ is cocomplete. In fact, this is a special case of the construction of a free cocomplete category generated by any category $\mathcal{C}$, given by the Yoneda embedding of $\mathcal{C}$ into the category $\mathbf{Set}^{\mathcal{C}^{op}}$ of presheaves on $\mathcal{C}$, see e.g. [Awo06] (Example 9.15 and Prop. 9.16). This also justifies the following choice of representation of the original signatures here: we identify Σ_1 with $\emptyset \to \{id_{\Sigma_1}\}$ and Σ_2 with $\{id_{\Sigma_2}\} \to \{\delta\}$, and δ with the unique morphism between them. We may write $\Sigma_\emptyset$ for the initial signature $\emptyset \to \emptyset$.

In $\mathbf{Set}^{\rightarrow}$, the signature $X_2 \xrightarrow{f} X_1$ is a colimit of a diagram with nodes $X_2 \uplus X_1$, where Σ_2 is the object in each node in X_2 and Σ_1 is the object in each node in X_1, with edges from $f(s)$ to s labelled by δ for each node $s \in X_2$. Now, extend the model functor so that the category $\mathbf{Mod}'_0(X_2 \xrightarrow{f} X_1)$ is the limit in **ICat** of the image of this diagram w.r.t. $\mathbf{Mod}_0$ (as defined above on Σ_1, Σ_2 and δ). This means in particular that a model P over a signature $X_2 \xrightarrow{f} X_1$ is a pair of functions $P_1 \colon X_1 \to \{K_1, N_1, M_1\}$ and $P_2 \colon X_2 \to \{N_2, M_2\}$ such that for $x_2 \in X_2$, $P_1(f(x_2)) = P_2(x_2)\big|_\delta$. Such a model P is a submodel of P' iff $P_2(x_2) = P'_2(x_2)$ for $x_2 \in X_2$, and $P_1(x_1) \subseteq P'_1(x_1)$ for $x_1 \in X_1$. Model reducts are given by (pre)composition with signature morphisms. Then for the sentence functor we set $\mathbf{Sen}_0(X_2 \xrightarrow{f} X_1) = \emptyset$.

Now, the institution so sketched is exact, and the counterexample works: the specification

$$\textbf{generate by } \sigma \textbf{ from } \langle \Sigma_\emptyset, \emptyset \rangle \textbf{ in } (\langle \Sigma_2, \emptyset \rangle \textbf{ hide via } \delta)$$

has no equivalent Σ_1-specification in basic normal form. To see this, just note that for any basic generating constraint C with $Sig[C] = (X_2 \xrightarrow{f} X_1)$, for any $P \in Mod[C]$, if for $x_1 \in (X_1 \setminus f(X_2))$, $P(x_1) = N_1$ then there are $P', P'' \in Mod[C]$ such that $P'(x_1) = K_1$ and $P''(x_1) = M_1$, and if for $x_2 \in X_2$, $P(x_2) = N_2$ then there is $P' \in Mod[C]$ such that $P'(x_2) = M_2$. Moreover, this property of specifications is preserved under translation w.r.t. any signature morphism. It follows then that no Σ_1-specification in basic normal form has N_1 (or rather, P such that $P_1(id_{\Sigma_1}) = N_1$) as its unique model. □

The source of the trouble is the use of hiding within the target specifications for generating constraints. One might suppose that when hiding is forbidden, the normal form result holds even if other specification-building operations are permitted within the target specifications used in generating constraints. Unfortunately, this is not the case: nested generating constraints may yield a similar effect as captured by Counterexample 6.5.

Counterexample 6.6. Consider a very simple institution $\mathbf{INS}_1$ with three signatures Σ, Σ_1 and Σ_2 and non-identity morphisms $\sigma_1 \colon \Sigma_1 \to \Sigma$ and $\sigma_2 \colon \Sigma_2 \to \Sigma$. $\mathbf{Mod}_1(\Sigma_1)$ has three distinct models K_1, LN_1 and M_1 with $K_1 \subseteq LN_1 \subseteq M_1$, $\mathbf{Mod}(\Sigma_2)$ has two distinct models KL_2 and NM_2 with $KL_2 \subseteq NM_2$, and $\mathbf{Mod}(\Sigma)$ has four distinct models K, L, N and M with $K \subseteq L \subseteq N \subseteq M$. We put $K\big|_{\sigma_1} = K_1$, $L\big|_{\sigma_1} = N\big|_{\sigma_1} = LN_1$ and $M\big|_{\sigma_1} = M_1$, and $K\big|_{\sigma_2} = L\big|_{\sigma_2} = KL_2$ and $N\big|_{\sigma_2} = M\big|_{\sigma_2} = NM_2$. Finally, we assume that there are no sentences in $\mathbf{INS}_1$, i.e., $\mathbf{Sen}_1(\Sigma) = \mathbf{Sen}_1(\Sigma_1) = \mathbf{Sen}_1(\Sigma_2) = \emptyset$.

Then $Mod[\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma, \emptyset \rangle] = \{K, L, M\}$, and so the constraint

$$\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } (\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } \langle \Sigma, \emptyset \rangle)$$

has K and M as its only models. It is easy to check though that no basic generating constraint, and no specification in basic normal form, has $\{K, M\}$ as its model class.

As in Counterexample 6.5, the above does not quite give a counterexample: the defined institution is not exact. Therefore, a construction of a new institution $\mathbf{INS}_1'$ analogous to that in Counterexample 6.5 has to be carried out.

Let $\mathcal{C}_1$ be the category with three signatures and morphisms as defined above. Take its free cocomplete closure via the Yoneda embedding into the category of presheaves over $\mathcal{C}_1$, $Y\colon \mathcal{C}_1 \to \mathbf{Set}^{\mathcal{C}_1^{op}}$. More explicitly, the resulting new category of signatures has objects of the form $X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2$, where X_1, X and X_2 are sets and $f_1\colon X \to X_1$ and $f_2\colon X \to X_2$ are functions. A morphism $h\colon (X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2) \to (X_1' \stackrel{f_1'}{\leftarrow} X' \stackrel{f_2'}{\rightarrow} X_2')$ consists of three functions $h_1\colon X_1 \to X_1'$, $h_0\colon X \to X'$ and $h_2\colon X_2 \to X_2'$ such that $h_0;f_1' = f_1;h_1$ and $h_0;f_2' = f_2;h_2$. We identify Σ_1 with $\{id_{\Sigma_1}\}\leftarrow\emptyset\rightarrow\emptyset$, Σ_2 with $\emptyset\leftarrow\emptyset\rightarrow\{id_{\Sigma_2}\}$, Σ with $\{\sigma_1\}\leftarrow\{id_\Sigma\}\rightarrow\{\sigma_2\}$, and the morphisms $\sigma_1\colon \Sigma_1 \to \Sigma$ and $\sigma_2\colon \Sigma_2 \to \Sigma$ with unique morphisms between them. We do not add any sentences, so that the sentence functor $\mathbf{Sen}_1'$ yields the empty set on every signature.

A signature $X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2$ is a colimit of a diagram with nodes $X_1 \uplus X \uplus X_2$, where nodes in X_1 carry Σ_1, nodes in X carry Σ and nodes in X_2 carry Σ_2, and edges from $f_1(x)$ to x are labelled by σ_1 and from $f_2(x)$ to x are labelled by σ_2, for all $x \in X$. We define the model functor $\mathbf{Mod}_1'$ so that $\mathbf{Mod}_1'(X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2)$ is the limit in $\mathbf{ICat}$ of the image of this diagram under $\mathbf{Mod}_1$ (as defined above for $\mathcal{C}_1$). That is, any model $P \in |\mathbf{Mod}_1'(X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2)|$ consist of three functions $P_1\colon X_1 \to \{K_1, LN_1, M_1\}$, $P_0\colon X \to \{K, L, N, M\}$ and $P_2\colon X_2 \to \{KL_2, NM_2\}$ such that for $x \in X$, $P_0(x)|_{\sigma_1} = P_1(f_1(x))$ and $P_0(x)|_{\sigma_2} = P_2(f_2(x))$. Such a model is a submodel of $P' \in |\mathbf{Mod}(X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2)|$ if for all $x_1 \in X_1$, $P_1(x_1) \subseteq P_1'(x_1)$, and similarly for X and X_2. Model reducts are given by (pre)composition with signature morphisms. For a class of models $\mathcal{P} \subseteq |\mathbf{Mod}_1'(X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2)|$ and $x \in X$, we write $\mathcal{P}(x) = \{P_0(x) \mid P \in \mathcal{P}\}$.

Consider now a signature morphism $h\colon (X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2) \to (X_1' \stackrel{f_1'}{\leftarrow} X' \stackrel{f_2'}{\rightarrow} X_2')$ and constraint $C' = \mathbf{generate\ by}\ h\ \mathbf{from}\ \langle X_1 \stackrel{f_1}{\leftarrow} X \stackrel{f_2}{\rightarrow} X_2, \emptyset\rangle\ \mathbf{in}\ \langle X_1' \stackrel{f_1'}{\leftarrow} X' \stackrel{f_2'}{\rightarrow} X_2', \emptyset\rangle$. We analyse the class of the models of C'. Let $x' \in X'$.

- For some $x \in X$, $x' = h_0(x)$ (and so $f_1'(x') = h_1(f_1(x))$ and $f_2'(x') = h_2(f_2(x))$). Then $Mod[C'](x') = \{K, L, N, M\}$, since informally, the corresponding component of the signature morphism is the identity on this "occurrence" of Σ.
- x' is not in the image of h_0; then we have the following subcases.
  - For some $x_1 \in X_1$ and $x_2 \in X_2$, $h_1(x_1) = f_1'(x')$ and $h_2(x_2) = f_2'(x')$. Then $Mod[C'](x') = \{K, L, N, M\}$, since informally, the corresponding component of the signature morphism is the map from the coproduct of Σ_1 and Σ_2 to this "occurrence" of Σ given by σ_1 and σ_2, and the reducts w.r.t. σ_1 and σ_2 do not jointly identify any models from $\{K, L, N, M\}$.
  - For some $x_1 \in X_1$, $h_1(x_1) = f_1'(x')$ but $f_2'(x')$ is not in the image of h_2. Then $Mod[C'](x') = \{K, L, M\}$, since informally, the corresponding

component of the signature morphism is σ_1, and the reduct w.r.t. σ_1 glues L and N together.

- For some $x_2 \in X_2$, $h_2(x_2) = f'_2(x')$ but $f'_1(x')$ is not in the image of h_1. Then $Mod[C'](x') = \{K, N\}$, since informally, the corresponding component of the signature morphism is σ_2, and the reduct w.r.t. σ_2 glues K and L as well as N and M together.
- Neither is $f'_1(x')$ in the image of h_1 nor is $f'_2(x')$ in the image of h_2. Then $Mod[C'](x') = \{K\}$, since informally, the corresponding component of the signature morphism is the unique morphism from the initial signature to Σ, and the reduct w.r.t. this morphism glues all models together.

Consequently, for any basic generation constraint C' as above, for $x' \in X'$, the class $Mod[C](x')$ is in the family $\mathcal{F} = \{\{K, L, N, M\}, \{K, L, M\}, \{K, N\}, \{K\}\}$. Moreover, this property is preserved under translation of specifications, since the family $\mathcal{F}$ is closed under intersection, and under hiding (reducts w.r.t. signature morphisms). Therefore, no specification in basic normal form may have $\{K, M\}$ is its class of models. □

The above counterexamples show that in general we cannot avoid nesting of structured specifications within generation constraints. We say that a specification is in *nested normal form* if either it is a basic specification, or it is built as follows:

$$((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } SP') \textbf{ with } \sigma') \textbf{ hide via } \delta$$

where SP' is a specification in nested normal form.

Corollary 6.7. *In any finitely exact institution* **INS** *with model inclusions, any structured specification built from basic specifications using union, translation, hiding and generating constraints is equivalent to a specification in nested normal form.*

Proof. As in the proof of Thm. 6.4, we first use Cor. 6.3 to allow us to deal with source-trivial generating constraints only. Then the proof proceeds by double induction, on the maximal depth of nesting of generating constraints in the specifications, and then on the structure of specifications. When the depth of nesting is at most 1, the result follows by Thm. 6.4. Otherwise, we proceed by induction on the structure of specification, assuming the thesis for all specifications with a smaller depth of nesting of generating constraints. The case of basic specifications is trivial. The cases for translation and hiding follow much as in the proof of Thm. 6.4. For generating constraints, the thesis follows by the inductive assumption, since the specification used within the generating constraint has a smaller depth of nesting of generating constraints. For the case of union, we get by an argument analogous to that in the proof of Thm. 6.4 for the case

of union:

$$
\begin{array}{l}
(((\textbf{generate by } \sigma_1 \textbf{ from } \langle \Sigma_1, \emptyset \rangle \textbf{ in } SP'_1) \textbf{ with } \sigma'_1) \textbf{ hide via } \delta_1) \\
\cup \\
(((\textbf{generate by } \sigma_2 \textbf{ from } \langle \Sigma_2, \emptyset \rangle \textbf{ in } SP'_2) \textbf{ with } \sigma'_2) \textbf{ hide via } \delta_2) \\
\quad \equiv \\
\big(\big((\textbf{generate by } \sigma_0 \textbf{ from } \langle \Sigma_0, \emptyset \rangle \textbf{ in } ((SP'_1 \textbf{ with } i'_1) \cup (SP'_2 \textbf{ with } i'_2)))\big) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \textbf{with } \sigma'_0\big) \textbf{ hide via } \delta_0
\end{array}
$$

Now, the thesis follows by the inductive assumption, since the depth of nesting of generating constraints in $(SP'_1 \textbf{ with } i'_1) \cup (SP'_2 \textbf{ with } i'_2)$ *is lower than in the original specification.* □

7 Final remarks

We started with the normal form result for constraints as studied in [EWT83] in the standard algebraic framework. We have shown that this result carries over to the more general setting of an arbitrary institution with some minimal extra structure: the notion of a submodel needed to capture the definition of generated model used in [EWT83]. Moreover, we sharpened the result somewhat via the use of a more restrictive definition of normal form.

We then considered the more general problem of normalising specifications where generating constraints are imposed in a class of models of an arbitrary specification, not just a presentation as in [EWT83]. Unfortunately, two counterexamples show that the normal form result does not carry over to this more general situation. Some nesting of generating constraints must be allowed, leading to a considerably weaker normal form result for this more general case.

The difficulties we encountered are linked to the definition of generated model in [EWT83], which we retained here. A standard alternative would be to free the concept of generated model from its dependency on the class of models of the specification at hand, and consider generation in the class of all models over the given signature. In the standard algebraic framework this leads to the usual notion of generated algebra, where all elements are values of terms with variables taking values in the indicated carriers, with the usual connection to structural induction, as in CASL [BCH+04]. For specifications with generating constraints of this special form, by easy adaptation of Thm. 6.4 and its proof one can build an equivalent normal form of the following shape:

$$(((\textbf{generate by } \sigma \textbf{ from } \langle \Sigma, \emptyset \rangle \textbf{ in } \langle \Sigma', \emptyset \rangle) \textbf{ then } \Phi) \textbf{ with } \sigma') \textbf{ hide via } \delta$$

Restricting to this special case would considerably limit the power of generating constraints as considered here. For instance, in AI applications, McCarthy's notion of circumscription [McC80] used to impose a "closed world assumption" could not be captured in general, since no predicate ever holds in generated models over a first-order signature without any axioms or constraints imposed

on the class of models considered. It is worth mentioning that a similarly general construct was introduced in DOL, the Distributed Ontology, Modeling and Specification Language [MCNK15].

One issue we did not touch on here at all is the development of proof systems for structured specifications. This is well-studied in the context of specifications built from basic specifications using union, translation and hiding, with a standard compositional proof system for consequences of specifications given in the framework of an arbitrary institution already in [ST88]. Completeness results follow under additional assumptions about the institution (most notably, interpolation is needed) where the proof of completeness heavily relies on the normal form result [Bor02]. It is well-known that once generating constraints are added, there is no hope for completeness [MS85]. However, in [ST14] we showed that the compositional proof system for structured specification built from basic specifications using union, translation and hiding is the best sound compositional proof system possible. It would be interesting to see how to carry this over to specifications with generating constraints, with some sound approximate techniques for proving consequences of generating constraints. Perhaps the normal form results studied here could be used to "concentrate" the necessary incompleteness at specific points in the structure of specifications, linked to the use of generating constraints in the normal forms.

Acknowledgements: Thanks to the anonymous referees for their constructive comments.

References

[Awo06] Steve Awodey. *Category Theory*. Oxford University Press, 2006.

[BCH+04] Hubert Baumeister, Maura Cerioli, Anne Haxthausen, Till Mossakowski, Peter D. Mosses, Donald Sannella, and Andrzej Tarlecki. CASL semantics. In [Mos04]. 2004.

[BHK90] Jan A. Bergstra, Jan Heering, and Paul Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.

[BM04] Michel Bidoit and Peter D. Mosses, editors. CASL *User Manual*, volume 2900 of *Lecture Notes in Computer Science*. Springer, 2004. See also `http://www.informatik.uni-bremen.de/cofi/index.php/CASL`.

[Bor02] Tomasz Borzyszkowski. Logical systems for structured specifications. *Theoretical Computer Science*, 286(2):197–245, 2002.

[CMST17] Mihai Codescu, Till Mossakowski, Donald Sannella, and Andrzej Tarlecki. Specification refinements: calculi, tools, and applications. *Science of Computer Programming*, 144:1–49, 2017.

[CR97] Virgil Emil Căzănescu and Grigore Roşu. Weak inclusion systems. *Mathematical Structures in Computer Science*, 7(2):195–206, 1997.

[DGS93] Răzvan Diaconescu, Joseph A. Goguen, and Petros Stefaneas. Logical support for modularisation. In Gérard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge University Press, 1993.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of Algebraic Specification 1*, volume 6 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1985.

[EWT83] Hartmut Ehrig, Eric G. Wagner, and James W. Thatcher. Algebraic specifications with generating constraints. In *Proceedings of the 10th International Colloquium on Automata, Languages and Programming*, volume 154 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 1983.

[GB84] Joseph A. Goguen and Rodney M. Burstall. Introducing institutions. In *Proceedings of the Workshop on Logics of Programs*, volume 164 of *Lecture Notes in Computer Science*, pages 221–256. Springer, 1984. Many revised versions were widely circulated, with [GB92] as the endpoint.

[GB92] Joseph A. Goguen and Rodney M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[GR04] Joseph A. Goguen and Grigore Roşu. Composing hidden information modules over inclusive institutions. In *From Object-Orientation to Formal Methods. Essays in Memory of Ole-Johan Dahl*, volume 2635 of *Lecture Notes in Computer Science*, pages 96–123. Springer, 2004.

[McC80] John McCarthy. Circumscription — A form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.

[MCNK15] Till Mossakowski, Mihai Codescu, Fabian Neuhaus, and Oliver Kutz. The Distributed Ontology, Modeling and Specification Language — DOL. In Arnold Koslow and Arthur Buchsbaum, editors, *The Road to Universal Logic*, volume 2, pages 489–520. Birkhäuser, 2015.

[Mos04] Peter D. Mosses, editor. CASL *Reference Manual*, volume 2960 of *Lecture Notes in Computer Science*. Springer, 2004.

[MS85] David MacQueen and Donald Sannella. Completeness of proof systems for equational specifications. *IEEE Transactions on Software Engineering*, SE-11(5):454–461, 1985.

[ST88] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76(2–3):165–210, 1988.

[ST12] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012.

[ST14] D. Sannella and A. Tarlecki. Property-oriented semantics of structured specifications. *Mathematical Structures in Computer Science*, 24(2):e240205, 2014.

[SW83] Donald Sannella and Martin Wirsing. A kernel language for algebraic specification and implementation. In *Proceedings of the 1983 International Conference on Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 1983.

[Tar83] Andrzej Tarlecki. Remarks on "Algebraic Specifications with Generation Constraints" by H. Ehrig, E.G. Wagner, J.W. Thatcher (ICALP'83, LNCS 154, 188-202). Unpublished note, Dept. of Computer Science, University of Edinburgh, 1983.