

Colimits of Order-Sorted Specifications

Till Mossakowski

Department of Computer Science,
University of Bremen, P.O.Box 33 04 40, D-28334 Bremen, Germany,
E-mail till@informatik.uni-bremen.de

Abstract. We prove cocompleteness of the category of CASL signatures, of monotone signatures, of strongly regular signatures and of strongly locally filtered signatures. This shows that using these signature categories is compatible with a pushout or colimit based module system.

1 Introduction

“Given a species of structure, say widgets, then the result of interconnecting a system of widgets to form a super-widget corresponds to taking the *colimit* of the diagram of widgets in which the morphisms show how they are interconnected.” J. Goguen [8]

An important application of this is the slogan “Putting theories together to make specifications” [3]. That is, specifications should be developed in a modular way, using colimits to combine different modules properly.

An orthogonal question is that of the *logic* that is used to specify the individual modules. Order-sorted algebra is a logic that has been proposed as a means to deal with exceptions, partiality and inheritance. See, among others, Goguen and Meseguer’s survey paper [10]. The specification languages OBJ3 [11], CafeOBJ [7] and CASL [15, 4] are all based on (extensions of) order-sorted algebra.

Now the combination (or should we say “pushout”?) of the two previous paragraphs would be an approach to order-sorted algebra where signatures and theories indeed can be combined with colimits. Surprisingly, such an approach was proposed only recently by Haxthausen and Nickl [12]. They examine how conditions like regularity or local filtration proposed by [10] interact with pushouts. They prove the existence of pushouts of signatures which are constrained by these conditions. Unfortunately, the existence of pushouts is guaranteed only under some severe restrictions. In particular, all signature morphisms have to be embeddings.

In this work, we propose an alternative way to deal with colimits of order-sorted specifications. Rather than trying to find restrictive conditions under which pushouts of order-sorted signatures preserve properties like monotonicity, regularity or local filtration, we change the perspective and allow the pushout of regular signatures to differ from the pushout of ordinary signatures. Technically,

we drop the implicit assumption of [12] that the functor forgetting regularity (or other conditions) should preserve colimits.

The main result states that the categories of monotone, of (strongly) regular and of (strongly) locally filtered signatures, resp., together with morphisms which preserve in some natural way the monotone, (strongly) regular and (strongly) locally filtered structure, resp., all are cocomplete. Thus we can combine order-sorted signatures in an unrestricted way.

In this paper, we choose the subsorting approach of CASL, since CASL does not impose any condition on order-sorted signatures and thus seems to be the most general approach. The paper is organized as follows: We introduce some preliminaries about subsorting in CASL and some category theory results in Sect. 2. The following sections deal with pure subsorting (without any conditions), monotonicity, partial orderedness, regularity and local filtration, resp. In particular, in each section a cocompleteness theorem is proved. Sect. 8 contains the conclusions.

2 Preliminaries

In this section we first recall the CASL approach to order-sorted algebra, and then some category theory that will be useful for dealing with colimits.

2.1 Subsorting in CASL

CASL stands for ‘‘Common Algebraic Specification Language’’ and is the central language within the family of CoFI languages. CoFI¹ is an initiative to design *a Common Framework for Algebraic Specification and Development* [15].

We are interested in CASL here since it has a new and very general approach to subsorting, which is combined with overloading and partiality.

We first recall the notion of CASL-signature and signature morphism from [5, 4], using a slightly different but equivalent formulation.

Definition 2.1. A *CASL subsorted signature* (S, F, TF, P, \leq_S) consists of

- a set S of *sorts*
- an $S^* \times S$ -sorted family $F = (F_{w,s})_{w \in S^*, s \in S}$ of *function symbols*,
- an $S^* \times S$ -sorted family $TF = (TF_{w,s} \subseteq F_{w,s})_{w \in S^*, s \in S}$ of subsets indicating the *total function symbols*,
- an S^* -sorted family $P = (P_w)_{w \in S^*}$ of *predicate symbols* and
- a pre-order (i.e. a reflexive transitive relation) \leq_S of *subsort embeddings* on the set S of sorts. □

Notation: The relation \leq_S naturally extends to sequences of sorts. We drop the subscript S when obvious from the context. We write $f: w \rightarrow s \in F$ for $f \in F_{w,s}$ (even if f is a partial function symbol, which is denoted by $f: w \rightarrow? s$)

¹ CoFI is an acronym for *Common Framework Initiative* and is pronounced like ‘coffee’.

in CASL specifications) and $p : w$ for $p \in P_w$. $f : w \rightarrow s \in F$ and $p : P_w$ are called *profiles*, w is called the *arity* and s the *coarity*.

Note that the signatures are not required to be monotonic, regular or locally filtered as in [10]. This decision was taken in order both to allow arbitrary overloading and to get a cocompleteness theorem (see Theorem 3.1 below).

For a subsorted signature $\Sigma = (S, F, TF, P, \leq_S)$, we define *overloading relations* (also called *monotonicity orderings*), \sim_F and \sim_P , for function and predicate symbols, respectively:

Definition 2.2. For $f : w_1 \rightarrow s_1, f : w_2 \rightarrow s_2 \in F, f : w_1 \rightarrow s_1 \sim_F f : w_2 \rightarrow s_2$ iff there exist $w \in S^*$ with $w \leq w_1, w_2$ and $s \in S$ with $s \geq s_1, s_2$.

For $p : w_1, p : w_2 \in P, p : w_1 \sim_P p : w_2$ iff there exists $w \in S^*$ with $w \leq w_1, w_2$.

These overloading relations lead to semantical identifications of different profiles in the models.

Definition 2.3. Given signatures $\Sigma = (S, F, TF, P, \leq)$ and $\Sigma' = (S', F', TF', P', \leq')$, a signature morphism $\sigma : \Sigma \rightarrow \Sigma'$ consists of

- a map $\sigma^S : S \rightarrow S'$,
- a map $\sigma_{w,s}^F : F_{w,s} \rightarrow F'_{\sigma^{S^*}(w), \sigma^S(s)}$ ² for each $w \in S^*, s \in S$ and
- a map $\sigma_w^P : P_w \rightarrow P'_{\sigma^{S^*}(w)}$ for each $w \in S^*$

such that

- subsorting is preserved, i. e. $s_1 \leq s_2$ implies $\sigma^S(s_1) \leq \sigma^S(s_2)$ for $s_1, s_2 \in S$,
- totality is preserved, i. e. $\sigma_{w,s}^F(TF_{w,s}) \subseteq TF'_{\sigma^{S^*}(w), \sigma^S(s)}$,
- the overloading relations are preserved³, i. e. $f : w_1 \rightarrow s_1 \sim_F f : w_2 \rightarrow s_2$ implies $\sigma_{w_1, s_1}^F(f) : \sigma^{S^*}(w_1) \rightarrow \sigma^S(s_1) \sim_F \sigma_{w_2, s_2}^F(f) : \sigma^{S^*}(w_2) \rightarrow \sigma^S(s_2)$ and $p : w_1 \sim_P p : w_2$ implies $\sigma_{w_1}^P(p) : \sigma^{S^*}(w_1) \sim_P \sigma_{w_2}^P(p) : \sigma^{S^*}(w_2)$.

Note that, due to preservation of subsorting, the last condition can be simplified to: $f : w_1 \rightarrow s_1 \sim_F f : w_2 \rightarrow s_2$ implies $\sigma_{w_1, s_1}^F(f) = \sigma_{w_2, s_2}^F(f)$ (and similarly for the predicate symbols).

This gives us a category **CASLSig** of subsorted CASL signatures and signature morphisms. \square

Since we will later on specify the CASL signature category in CASL itself, we now give a brief overview over basic specifications in CASL. A basic specification in CASL consists of a list of basic items. A basic item may either be a declaration of some sorts, subsorts, partial or total functions or predicates, or be an axiom. Axioms are usual first-order formulae over the following atomic formulae:

² σ^{S^*} is the extension of σ^S to finite strings

³ The purpose of preservation of overloading is the following: the overloading relations lead to semantical identifications in the models. These are preserved by taking reducts only if the overloading relations are preserved by signature morphisms.

- Existential ($\stackrel{\varepsilon}{=}$) or strong ($=$) equalities between terms,
- applications of predicates to a list of terms,
- definedness tests (*defined* t), which check if a term is defined, or
- membership tests ($t \in s$), which check if a term belongs to a subsort.

As an example, consider the specification of **CASLSig** within CASL itself given in Fig. 1.

```

spec ORDEREDSTRINGS =
  sorts     $S, S^*$ 
  preds     $\_ \leq \_ : S \times S$ 
              $\_ \leq \_ : S^* \times S^*$ 
  ops       $\lambda : S^*$ 
              $\_ \_ : S \times S^* \rightarrow S^*$ 
  vars      $s, s_1, s_2, s_3 : S; w_1, w_2 : S^*$ 
  axioms    $s \leq s$ 
              $s_1 \leq s_2 \wedge s_2 \leq s_3 \Rightarrow s_1 \leq s_3$ 
              $\lambda \leq \lambda$ 
              $w_1 \leq w_2 \wedge s_1 \leq s_2 \Leftrightarrow s_1 w_1 \leq s_2 w_2$ 
              $s_1 w_1 = s_2 w_2 \Rightarrow (s_1 = s_2 \wedge w_1 = w_2)$ 
              $\lambda = sw \Rightarrow (s_1 = s_2 \wedge w_1 = w_2)$ 

spec CASLSIG =
  ORDEREDSTRINGS then
  sorts     FunProfiles, PredProfiles
  ops      arity: FunProfiles  $\rightarrow S^*$ 
             coarity: FunProfiles  $\rightarrow S$ 
             arity: PredProfiles  $\rightarrow S^*$ 
  preds    istotal : FunProfiles
              $\_ \sim_F \_ : FunProfiles \times FunProfiles$ 
              $\_ \sim_P \_ : PredProfiles \times PredProfiles$ 
  vars     fp1, fp2 : FunProfiles; pp1, pp2 : PredProfiles; s : S; w : S*
  axioms   (fp1  $\sim_F$  fp2  $\wedge$  arity(fp1) = arity(fp2)
              $\wedge$  coarity(fp1) = coarity(fp2))
              $\Rightarrow fp_1 = fp_2$ 
             (pp1  $\sim_P$  pp2  $\wedge$  arity(pp1) = arity(pp2))  $\Rightarrow fp_1 = fp_2$ 
              $\lambda = sw \Rightarrow (fp_1 = fp_2 \wedge pp_1 = pp_2 \wedge istotal(fp_1))$ 

```

Figure 1. A specification of the CASL signature category within CASL

Subsorted partial-first order logic (*SubPFOL*) is the underlying institution of CASL. It is described in [6, 4]. For our cocompleteness proofs, we will need the following proposition, which follows from Corollaries 15 and 17 in [13].

Proposition 2.4. *Let $\theta: T \rightarrow T1$ be a theory morphism in SubPFOL between universal Horn theories which do not use subsorting and which do use strong*

equalities only in the conclusions of Horn formulae.⁴

Then both $\mathbf{Mod}(T)$ and $\mathbf{Mod}(T1)$ are cocomplete, and the forgetful functor $-\downarrow_{\emptyset}: \mathbf{Mod}(T1) \rightarrow \mathbf{Mod}(T)$ has a left adjoint.⁵ \square

2.2 Some categorical tools for proving cocompleteness

In this section, we recall some results from category theory that comprise a useful toolkit for proving cocompleteness theorems.

Definition 2.5 ([1], 13.17). A functor $F: \underline{A} \rightarrow \underline{B}$ is said to *lift colimits*, if for every diagram $D: \underline{I} \rightarrow \underline{A}$ and every colimit \mathcal{C} of $F \circ D$ there exists a colimit \mathcal{C}' of D with $F(\mathcal{C}') = \mathcal{C}$. \square

Definition 2.6 ([1], 4.16). Let \underline{A} be a subcategory of \underline{B} , and let B be a \underline{B} -object. An \underline{A} -*reflection arrow* for B is a morphism $r: B \rightarrow A$ into an \underline{A} -object A with the following universal property:

For any morphism $f: B \rightarrow A'$ from B into some \underline{A} -object A' , there exists a unique \underline{A} -morphism $f': A \rightarrow A'$ such that $f = f' \circ r$.

\underline{A} is called a *reflective subcategory* of \underline{B} provided that each \underline{B} -object has an \underline{A} -reflection.⁶

The dual notion is that of *co-reflective* subcategory. \square

Proposition 2.7 ([2], 3.5). If \underline{A} is a full, reflective or coreflective, subcategory of \underline{B} , and \underline{B} is cocomplete, then \underline{A} is cocomplete as well. \square

Proposition 2.8. Consider a commuting diagram of four subcategories

$$\begin{array}{ccc}
 \underline{A} & \xrightarrow{\text{full}} & \underline{A}' \\
 \uparrow & & \uparrow \\
 & \text{reflective} & \\
 \underline{B} & \xrightarrow{\text{full}} & \underline{B}'
 \end{array}$$

where \underline{A} is a full subcategory of \underline{A}' , \underline{B} is a full subcategory of \underline{B}' and \underline{B}' is a reflective subcategory of \underline{A}' .

If the \underline{B}' -reflection of an arbitrary \underline{A}' -object coming from \underline{A} is already a \underline{B} -object, then \underline{B} is a reflective subcategory of \underline{A} . \square

⁴ The proposition can also be proved without the restriction that subsorting is not used, but forbidding strong equations in the premises of the Horn formulae is essential.

⁵ For the purpose of this paper, we assume that empty carriers are allowed in the models of a CASL theory.

⁶ Note that this is equivalent to the condition that the inclusion functor from \underline{A} to \underline{B} has a left adjoint, which then produces the reflection.

3 Cocompleteness of the CASL signature category

For modularity aspects, it is important that signatures can be combined. In particular, CASL has the construct of generic extensions, which have a formal parameter part that can be later instantiated with different actual parameters. Instantiation of generic extensions is done via a construction which is a pushout. The existence of pushouts in CASL is guaranteed by the following:

Theorem 3.1. *The category **CASLSig** of subsorted CASL signatures and signature morphisms is cocomplete.*

Proof. Consider the universal Horn CASL specification **CASLSIG** given in Fig. 1. It might look a bit strange that we do not introduce a sort for names of functions and predicates, but rather for profiles. This is because we want to capture CASL signature morphisms by **CASLSIG**-homomorphisms. Now CASL signature morphisms can map different profiles for one and the same symbol name to profiles with different symbol names, while homomorphisms of course have only one choice for mapping elements of carriers. That's why profiles are appropriate as carriers, and not symbol names. Identity of names can be recovered by using the overloading relations. The axioms starting with $\lambda = sw$ ensure that any confusion of elements in S^* enforces the model to be terminal.

By Prop. 2.4 we know that $\mathbf{Mod}(\mathbf{CASLSIG})$, the model-category of **CASLSIG**, is cocomplete. The intention is now that **CASLSIG** specifies **CASLSig** somehow and thus **CASLSig** inherits cocompleteness.

Unfortunately, $\mathbf{Mod}(\mathbf{CASLSIG})$ is not equivalent to **CASLSig**: it is not guaranteed that there are no junk strings in the interpretation of S^* . Moreover, the profiles do not exactly capture function and predicate symbols. To repair this, we need the extension of **CASLSIG** given in Fig. 2.

```

spec CASLSIGGEN =
  CASLSIG then
  generated { sort       $S^*$ 
               ops       $\lambda : S^*$ 
                $-- \_ : S \times S^* \rightarrow S^*$  }
  vars       $s : S; w : S^*$ 
  axioms     $\neg (\lambda = sw)$ 
               $\neg (\lambda \leq sw)$ 
               $\neg (sw \leq \lambda)$ 
  vars       $fp_1, fp_2 : FunProfiles; pp_1, pp_2 : PredProfiles$ 
  axioms     $fp_1 \sim_F fp_2 \Rightarrow \exists w : S^*; s : S.$ 
               $(w \leq arity(fp_1) \wedge w \leq arity(fp_2))$ 
               $\wedge coarity(fp_1) \leq s \wedge coarity(fp_2) \leq s)$ 
               $pp_1 \sim_F pp_2 \Rightarrow \exists w : S^* \bullet (w \leq arity(pp_1) \wedge w \leq arity(pp_2))$ 

```

Figure 2. A specification of the CASL signature category within CASL

Lemma 3.2. $\mathbf{Mod}(\mathbf{CASLSIGGEN})$ is equivalent to $\mathbf{CASLSig}$.

Proof. A CASL-signature is mapped to a CASLSIGGEN-model by taking the obvious interpretations of the symbols. A CASLSIGGEN-model M is mapped to is the CASL-signature (M_S, F, TF, P, \leq_M) which consists of

- a set of sorts M_S (without loss of generality, $(M_S)^*$ can be identified with M_{S^*}),
- the pre-order \leq_M ,
- let $FunNames = M_{FunProfiles} / (\sim_F)_M$,
- for each $w \in (M_S)^*$ and $s \in M_S$, $F_{w,s} = \{c \in FunNames \mid \text{there is some } prof \in c \text{ with } arity_M(prof) = w, coarity_M(prof) = s\}$,
- for each $w \in (M_S)^*$ and $s \in M_S$, $c \in TF_{w,s}$ iff there is some $prof \in c$ with $arity_M(prof) = w$, $coarity_M(prof) = s$ and $istotal_M(prof)$,
- let $PredNames = M_{FunProfiles} / (\sim_P)_M$,
- for each $w \in (M_S)^*$, $P_{w,s} = \{c \in PredNames \mid \text{there is some } prof \in c \text{ with } arity_M(prof) = w\}$. \square

To prove cocompleteness of $\mathbf{Mod}(\mathbf{CASLSIGGEN})$, we cannot apply Prop. 2.4, since CASLSIGGEN is not in Universal Horn form. But we can apply the following lemma:

Lemma 3.3. $\mathbf{Mod}(\mathbf{CASLSIGGEN})$ is a coreflective subcategory of $\mathbf{Mod}(\mathbf{CASLSig})$.

Proof. The coreflection of a CASLSIG-model M that is not terminal is the following submodel M' of M :

- $M'_S = M_S$,
- M'_{S^*} is the subset of M_{S^*} generated by λ_M and $_{--}M$,
- \leq on M'_{S^*} is \leq on M_{S^*} restricted to arguments of same length (otherwise, it yields false),
- $FunProfiles_{M'} = \{prof \in FunProfiles_M \mid arity_M(prof) \in M'_{S^*}\}$,
- $prof_1 (\sim_F)_{M'} prof_2$ iff $prof_1 (\sim_F)_M prof_2$ and there exist some $w \in M'_{S^*}$ and $s \in M'_S$ with $w \leq_M arity_{M'}(prof_1)$, $w \leq_M arity_{M'}(prof_2)$, $coarity_{M'}(prof_1) \leq_M s$ and $coarity_{M'}(prof_2) \leq_M s$,
- $PredProfiles_{M'} = \{prof \in PredProfiles_M \mid arity_M(prof) \in M'_{S^*}\}$,
- $prof_1 (\sim_P)_{M'} prof_2$ iff $prof_1 (\sim_P)_M prof_2$ and there exists some $w \in M'_{S^*}$ with $w \leq_M arity_{M'}(prof_1)$ and $w \leq_M arity_{M'}(prof_2)$ and
- the other operations and relations are inherited from M .

The coreflection property can be seen as follows: Given a CASLSIGGEN-model $MGen$, a CASLSIG-model M and a CASLSIG-homomorphism $\sigma: MGen \rightarrow M$, σ is already a CASLSIG-homomorphism into the coreflection of M , since σ preserves both generatedness of elements of M_{S^*} and common super- and subsorts of arities and coarities, resp.

The coreflection of the terminal CASLSIG-model is the terminal CASLSIGGEN-model consisting of one sort and, for each number of arguments, one total function and one predicate profile. \square

Thus we can apply Prop. 2.7 to conclude that **CASLSig** is cocomplete as well. \square

Corollary 3.4. *The category of SubPFOL-theories is cocomplete.*

Proof. Cocompleteness of the signature category implies cocompleteness of the theory category for arbitrary institutions [9]. \square

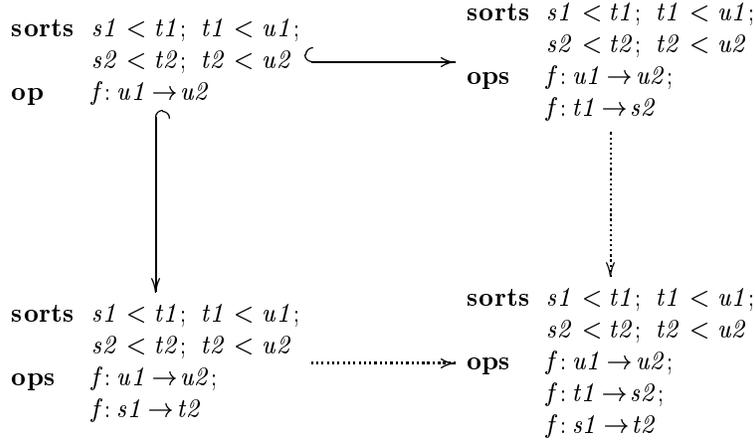
4 Monotonicity

Monotonicity is a condition that is assumed throughout the most popular survey paper about order-sorted algebra by Goguen and Meseguer [10]. A consequence of monotonicity is that overloading of constants is forbidden. More importantly, monotonicity is a preparing condition for regularity, which is discussed below.

Definition 4.1. A signature $\Sigma = (S, F, TF, P, \leq_s)$ is called *monotone*, if for $f : w_1 \rightarrow s_1$ and $f : w_2 \rightarrow s_2$, $w_1 \leq w_2$ implies $s_1 \leq s_2$. Let **MonSig** be the full subcategory of **CASLSig** consisting of monotone signatures. \square

Proposition 4.2 (Haxthausen and Nickl[12]). *The inclusion functor from **MonSig** to **CASLSig**⁷ does not lift colimits (in particular, it does not lift pushouts).* \square

This is illustrated by the following example:



The resulting signature is no longer monotone. That is, we cannot use the construction of colimits in **CASLSig** for constructing colimits in **MonSig** in

⁷ Haxthausen and Nickl use a category of signatures slightly different from **CASLSig**: they do not allow partial function and predicate symbols, they assume antisymmetry of the subsorting relation, and their preservation of overloading is stronger. But these differences do not matter here.

general. Now Haxthausen and Nickl study conditions under which pushouts are lifted from **CASLSig** to **MonSig**. These conditions are rather technical and restrictive. The signature morphisms used in the pushout have to be embeddings, which rules out instantiations of generics where different parts of the formal parameter are instantiated in the same way.

But there is a different way to get pushouts in **MonSig**. Rather than trying to lift colimits from **CASLSig** to **MonSig**, we can try to modify **CASLSig**-colimits in order to get **MonSig**-colimits.

Proposition 4.3. *MonSig is a full reflective subcategory of CASLSig.*

Corollary 4.4. *MonSig is cocomplete.*

Proof. Given a CASL signature $\Sigma \in \mathbf{CASLSig}$, we construct its reflection $r: \Sigma \rightarrow \mathbf{Monotonify}(\Sigma)$ as follows:

Let θ be the inclusion of **CASLSig** into the theory

```
spec MONSIG =
  CASLSIG then
    vars   pr1, pr2 : FunProfile
    axiom  pr1 ~F pr2 ∧ arity(pr1) ≤ arity(pr2) ⇒ coarity(pr1) ≤ coarity(pr2)
```

By Prop. 2.4, **Mod**(MONSIG) is a full reflective subcategory of **Mod**(CASLSIG). Moreover, the reflection of a model induced by a CASL signature again is a model induced by a CASL signature, which is monotone. Thus by Prop. 2.8, **MonSig** is a reflective subcategory of **CASLSig**, and $\mathbf{Monotonify}(\Sigma)$ is obtained by first viewing Σ as a **CASLSig**-model and then constructing its free θ -extension. \square

For the above example, the reflection that gives the pushout in **MonSig** looks as follows:

<pre>sorts s1 < t1; t1 < u1; s2 < t2; t2 < u2 ops f: u1 → u2; f: t1 → s2; f: s1 → t2</pre>	\dashrightarrow	<pre>sorts s1 < t1; t1 < u1; s2 < t2; t2 < u2; t2 < s2 ops f: u1 → u2; f: t1 → s2; f: s1 → t2</pre>
--	-------------------	--

That is, the necessary subsort relationship leading to monotonicity is added.

5 Pre-ordered subsorts versus partially ordered subsorts

In CASL, the subsort relation on the set of sorts is a *pre-order*, that is, it may have sort cycles and thus fail to satisfy the law of antisymmetry:

$$s \leq s' \wedge s' \leq s \Rightarrow s = s'$$

In the models, a sort cycle leads to an isomorphism of all the carrier sets that correspond to sorts in the cycle. This possibility was consciously included into the design of CASL.

Now there are approaches to subsorting where the subsorting relation is required to be a *partial order*, and thus satisfy the law of antisymmetry. If one wants to feed a CASL specification into a tool which expect a partially-ordered set of sorts, one first has to quotient out the sort cycles. This can be done with the following:

Proposition 5.1. *Let \mathbf{PoSig} be the full subcategory of $\mathbf{CASLSig}$ determined by the signatures whose subsorting relation satisfy the law of antisymmetry. Then \mathbf{PoSig} is a reflective subcategory of $\mathbf{CASLSig}$.*

Corollary 5.2. *\mathbf{PoSig} is cocomplete.*

Proof. Let θ be the inclusion of $\mathbf{CASLSig}$ into the theory

```
spec  PoSig =
      CASLSig then
      vars   s1, s2 : S
      axiom  s1 ≤ s2 ∧ s2 ≤ s1 ⇒ s1 = s2
```

The rest of the proof parallels the proof of Prop. 4.3. □

$\mathbf{PoMonSig}$ denotes the intersection of \mathbf{PoSig} and \mathbf{MonSig} consisting of all partially ordered monotone signatures. Cocompleteness of this category follows with analogous arguments.

6 Regularity

Regularity is a property of subsorted signatures which leads to an extremely simple parsing algorithm which does the overload resolution.

Definition 6.1. A signature is *regular* if

- it is monotone and
- for every $f : w \rightarrow s$ and $w_0 \leq w$, there is a least profile for f with arity $\geq w_0$, and
- for every $p : w$ and $w_0 \leq w$, there is a least profile for p with arity $\geq w_0$.
- Moreover, we assume that the subsort relation is a partial order⁸.

Let $\mathbf{PoRegSig}$ be the (non-full) subcategory of \mathbf{PoSig} consisting of regular signatures and signature morphisms preserving the least profiles that are guaranteed by regularity. □

⁸ Throughout this section, we assume that all signatures have a subsort relation that is a partial order. In principle, the definitions and results of this section can be generalized to arbitrary pre-orders. However, the construction *Regularify* defined below gives satisfactory results only for partial orders (otherwise, too many new sorts are introduced). Note that a pre-ordered subsort relation can always be turned into a partially-ordered one, using the result of the previous section.

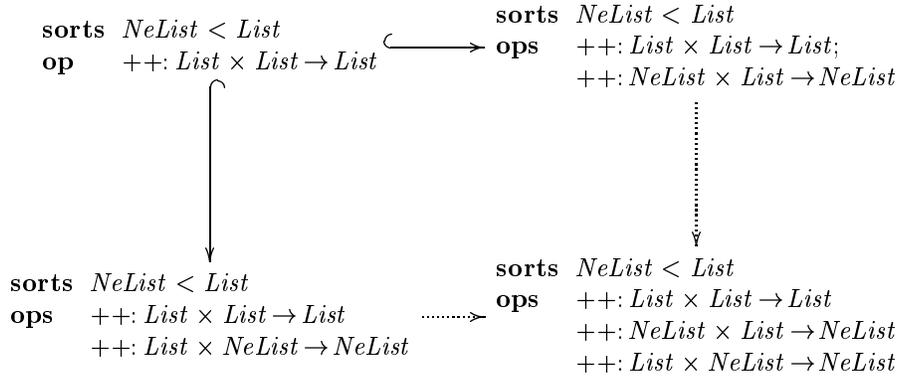
The main thing we gain from regularity is the existence of a simple bottom-up least sort parse algorithm due to Goguen and Meseguer [10]. To parse a term $f(t_1, \dots, t_n)$, recursively parse the t_i , giving a least sort parse $u_i : s_i$, and let $w_0 = s_1 \dots s_n$. If there is no $f: w \rightarrow s$ satisfying $w_0 \leq w$ in the signature, the term is ill-sorted. Otherwise, take the least such $f: w \rightarrow s$, which exists by regularity. The least sort parse of $f(t_1, \dots, t_n)$ is then $(f: w \rightarrow s)(u_1 : s_1, \dots, u_n : s_n) : s$.

Without regularity, we have to consider all possible sorts and parses of a given unparsed term or use a more complicated overload resolution algorithm [14].

A drawback of regularity is that it does not behave so well w. r. t. modularity:

Proposition 6.2 (Haxthausen and Nickl[12]). *The inclusion functor from **RegSig** to **CASLSig** does not lift colimits (in particular, it does not lift pushouts).* \square

This can be illustrated with the following example:



The three signatures forming the diagram are regular, but the pushout signature is not regular: if we take $w_0 = NeList \times NeList$, there is no least profile for $++$ with arity $\geq w_0$.

Again, Haxthausen and Nickl examine rather technical restrictions of pushout diagrams which guarantee that the pushout is lifted to **PoRegSig**. In particular, they require the signature morphisms to be embeddings. But as in the case of monotonicity, instead of trying to lift colimits, we can try to modify them by using a full reflective embedding.

Unfortunately, **PoRegSig** is not a full subcategory of **PoSig** (even though it is reflective). Therefore, we introduce the following restriction of regularity:

Definition 6.3. A monotone CASL-signature in **PoMonSig** is called *strongly regular*, if

- for every $f : w \rightarrow s$ and $w_0 \leq w$, w_0 is already an arity for f , that is, there is some s_0 with $f: w_0 \rightarrow s_0$, and

– for every $p : w$ and $w_0 \leq w$, w_0 is already an arity for p , that is, $p : w_0$.

This gives us a subcategory **PoStrRegSig** of **PoMonSig**. □

Corollary 6.4. *Every strongly regular signature is regular.* □

The restriction to strongly regular signatures is not as harmful as it may look in the first place. If we want to make a regular signature strongly regular, we just have to add some new profiles which can be interpreted by restricting interpretations of old profiles. Moreover, **PoStrRegSig** is a full⁹ subcategory of **PoRegSig** by the following proposition:

Proposition 6.5. *A signature morphism in **PoMonSig** starting from a strongly regular signature preserves the least profiles that are required to exist by regularity.*

Proof. Let $\sigma : \Sigma \rightarrow \Sigma'$ be a signature morphism in **PoMonSig** and Σ strongly regular. Let $f : w \rightarrow s$ and $w_0 \leq w$ in Σ . By strong regularity, the least profile for f with arity $\geq w_0$ has the form $f : w_0 \rightarrow s_0$. By preservation of overloading, $\sigma_{w,s}^F(f) = \sigma_{w_0,s_0}^F(f) = f'$. So by monotonicity, $f' : \sigma^S(w_0) \rightarrow \sigma^S(s_0)$ is the least profile for f' with arity $\geq \sigma^S(w_0)$ in Σ' . □

Proposition 6.6. *There is a functor *Regularify*: **PoSig** \rightarrow **PoStrRegSig** making **PoStrRegSig** a reflective subcategory of **PoSig**.*

Corollary 6.7. ***PoStrRegSig** is cocomplete.*

Proof. Let θ be the inclusion of **POSIG** into the theory

```

spec  POSTRREGSIG =
  POMONSIG then
  op    new_profile : FunProfiles  $\times$   $S^*$   $\rightarrow?$  FunProfiles
  vars   $w_0 : S^*$ ;  $f : FunProfile$ 
  axioms defined new_profile( $f, w_0$ )  $\Leftrightarrow w_0 \leq \text{arity}(f)$ 
          defined new_profile( $f, w_0$ )  $\Rightarrow \text{arity}(\text{new\_profile}(f, w_0)) = w_0$ 
          defined new_profile( $f, w_0$ )  $\Rightarrow f \sim_F \text{new\_profile}(f, w_0)$ 
  op    new_profilePredProfiles  $\times$   $S^*$   $\rightarrow?$  PredProfiles
  vars   $p : PreProfile$ 
          defined new_profile( $p, w_0$ )  $\Leftrightarrow w_0 \leq \text{arity}(p)$ 
          defined new_profile( $p, w_0$ )  $\Rightarrow \text{arity}(\text{new\_profile}(p, w_0)) = w_0$ 
          defined new_profile( $p, w_0$ )  $\Rightarrow p \sim_P \text{new\_profile}(p, w_0)$ 

```

The partial function *new_profile* generates, for each existing profile and each string of sorts w_0 less than the arity of the profile, a profiles with arity w_0 , in order to make the signature strongly regular. Note that the coarity of the new profile is left unspecified; this may lead to the generation of new sorts.

The rest of the proof parallels the proof of Prop. 4.3. □

⁹ even reflective

Now let us come back to the example. *Regularify* adds a new sort $L \leq NeList$ and a profile $++: NeList \times NeList \rightarrow L$ to the above pushout signature, giving a strongly regular pushout signature in *PoRegSig*.

$$\begin{array}{ccc}
 \text{sorts } NeList < List & & \text{sorts } NeList < List; \\
 \text{ops } ++: List \times List \rightarrow List & \xrightarrow{\dots\dots\dots} & \text{ops } ++: List \times List \rightarrow List \\
 ++: NeList \times List \rightarrow NeList & & ++: NeList \times List \rightarrow NeList \\
 ++: List \times NeList \rightarrow NeList & & ++: List \times NeList \rightarrow NeList \\
 & & ++: NeList \times NeList \rightarrow L
 \end{array}$$

For practical purposes, one would rename L by $NeList$ after forming the pushout.

7 Local Filtration

Locally filtered signatures have the property that satisfaction is closed under isomorphism in the approach of Goguen and Meseguer [10]. Also, the reducibility to many-sorted algebra is guaranteed only in case of local filtration. In CASL, we have both properties without local filtration. However, in CASL well-formedness of $t_1 = t_2$ and $t_2 = t_3$ do not necessarily ensure well-formedness of $t_1 = t_3$, but in case of local filtration, this implication holds¹⁰.

Definition 7.1. A signature $\Sigma \in \mathbf{PoSig}$ is called *locally filtered*, if each pair of sorts that is connected (i. e. that is in the symmetric transitive closure of \leq) has a common upper bound. This gives us a full subcategory **LFiltSig** of **PoSig**. \square

Now like the other properties, local filtration does not behave so well w. r. t. modularity:

Proposition 7.2 (Haxthausen and Nickl[12]). *The inclusion functor from LFiltSig to CASLSig does not lift colimits (in particular, it does not lift pushouts).* \square

This can be illustrated with the following example:

$$\begin{array}{ccc}
 \text{sort } Elem & \hookrightarrow & \text{sorts } Elem < List \\
 \downarrow \text{rename } Elem \text{ by } Nat & & \downarrow \dots\dots\dots \\
 \text{sorts } Nat < Int & \xrightarrow{\dots\dots\dots} & \text{sorts } Nat < List; \\
 & & Nat < Int
 \end{array}$$

¹⁰ In CASL, the implication holds also without local filtration, if we use *sorted* equations.

The three signatures forming the diagram are locally filtered, but the pushout signature is not: the sorts *List* and *Int* are connected, but do not have a common upper bound.

Again, Haxthausen and Nickl study rather restrictive conditions which guarantee that colimits are lifted to **LFiltSig**.

We cannot here apply our reflection technique directly, but need the following sharpening of local filtration:

Definition 7.3. A partially ordered signature in **PoS**ig is called *strongly locally filtered*, if each pair of connected sorts has a *least* common upper bound. Let **SLFiltSig** be the (non-full) subcategory of **LFiltSig** consisting of strongly locally filtered signatures and morphisms preserving least common upper bounds of connected pairs. \square

Unfortunately, we cannot use a reflection technique here to prove cocompleteness: **SLFiltSig** is a reflective subcategory of **PoS**ig, but not full! But we can prove cocompleteness more directly:

Proposition 7.4. **SLFiltSig** is cocomplete.

Proof. Consider the CASL specification

```

spec SLFILTsig =
  POsig then
  op     $\_ \wedge \_ : S \times S \rightarrow? S$ 
  vars   $s, s_1, s_2 : S$ 
  axioms  $s_1 \leq s_2 \Leftrightarrow s_1 \wedge s_2 = s_2$ 
           $s \wedge s = s^{11}$ 
           $s_1 \wedge s_2 = s_2 \wedge s_1$ 
           $(s_1 \wedge s_2) \wedge s_3 = s_1 \wedge (s_2 \wedge s_3)$ 
          defined  $s_1 \wedge s_2 \wedge$  defined  $s_2 \wedge s_3 \Rightarrow$  defined  $s_1 \wedge s_3$ 

```

\wedge is an operation that satisfies the axioms for *sup*-semilattices, but it is a *partial* operation. Of course, any pair of sorts for which \wedge is defined is connected. The last axiom ensures that the converse is true as well (note that by the first axiom, $s_1 \leq s_2$ implies *defined* $s_1 \wedge s_2$). Thus the set of sorts consists of a number of connected components, each of which is a *sup*-semilattice. This is nothing else than local filtration.

The rest of the proof parallels that of Theorem 3.1. \square

How do pushouts in **SLFiltSig** look? A pushout in **SLFiltSig** is constructed by taking the pushout in **PoS**ig and then adding, for each (finite) set of connected sorts without least upper bound, a new sort which now serves as (the set's) least upper bound. In particular, for the above example, we get

¹¹ Note that $=$ denotes strong equality: both sides are undefined or both defined and equal.

$$\begin{array}{ccc}
\text{sort } Elem & \hookrightarrow & \text{sorts } Elem < List \\
\downarrow \text{rename } Elem \text{ by } Nat & & \downarrow \text{dotted} \\
\text{sorts } Nat < Int & \dashrightarrow & \begin{array}{l} \text{sorts } Nat < List; \\ Nat < Int; \\ Int < Int \cup List; \\ List < Int \cup List; \end{array}
\end{array}$$

We can also use the left adjoint (=free construction) along the theory inclusion from PoSig to SLFiltSig to get a reflection which makes a partially ordered signature strongly locally filtered. By a general categorical theorem, this reflection also preserves colimits. But note that, different from the monotone and regular case, SLFiltSig is a non-full subcategory, which implies that the reflection of strongly locally filtered signatures may not be the identity. Thus a reflection of a colimit in PoSig of a diagram of signatures in SLFiltSig is a colimit in SLFiltSig but a colimit of the *reflected* diagram, which need not be the original one.

8 Conclusion and Comparison With Related Work

We have shown that the CASL specification language (which includes a new and general approach to subsorting) has a cocomplete signature category. This is important to be able to form instantiations of generic extension as pushouts.

For restrictions to monotone, regular or locally filtered signatures, Haxthausen and Nickl [12] have shown that colimits (in particular, pushouts) do not lift to the restricted signature categories. We argue that the approach of Haxthausen and Nickl of imposing severe restrictions on signatures and morphisms is not appropriate in all practical cases: they require morphisms to be embeddings, but there are instantiations of generics with non-injective fitting morphisms.

Therefore, we have developed a different approach to pushouts (and colimits in general) for monotone, regular and locally filtered signatures. We have shown that appropriate subcategories of such signatures are reflective, which means that colimits in the subcategories can be obtained by taking the reflection of the colimit in the original category.

A short comparison of the advantages and disadvantages of the CASL approach (without any monotonicity, regularity and local filtration restriction) and of Haxthausen and Nickl's and our approach to colimits for regular signatures is given in the following table:

Approach	advantages	disadvantages
CASL[15]	conceptually very simple and general	overload resolution is involved
Haxthausen and Nickl[12]	easy parsing, literal pushouts	restricted to embeddings, technically involved
Reflections (our approach)	easy parsing, conceptually simple	New sorts and relations are added, amalgamation property may fail

The crucial feature of our approach is the generation of new profiles, subsort relationships and even sorts. The former are not so harmful (and even may be consider as helpful), and the new sorts generated for locally filtered signatures may be interpreted as sort unions. Only for regular signatures, pushouts may contain new sorts that cannot be interpreted in a useful way and rather should be renamed into existing sorts after forming the pushout.

The main advantage of our approach is the combination of the conceptual simplicity of the CASL approach with the possibility to use monotonicity, regularity and local filtration, e.g. in order to ease parsing.

Owe and Dahl [16] also describe an algorithm that makes a signature regular. It works with intersections and unions of types, assuming that there is a family of basic types with disjoint interpretation. To make the algorithm work, they have to assume that signatures are consistent, which means that some interpretation with non-empty carriers exists (in CASL, such an interpretation always exists, but in their approach, there is a disjointness requirement that may lead to the non-existence of such an interpretation). Now unfortunately consistency is a property that is not stable under pushouts. So Owe and Dahl’s regularifying algorithm, while interesting in general, does not help in the present setting.

Acknowledgements I would like to thank Anne Haxthausen and Olaf Owe for fruitful discussions and the anonymous referee for some useful hints.

References

1. J. Adámek, H. Herrlich, and G. Strecker. *Abstract and Concrete Categories*. Wiley, New York, 1990.
2. F. Borceux. *Handbook of Categorical Algebra I – III*. Cambridge University Press, 1994.
3. R. M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 1045–1058. Cambridge, 1977.
4. M. Cerioli, A. Haxthausen, B. Krieg-Brückner, and T. Mossakowski. Permissive subsorted partial logic in CASL. In M. Johnson, editor, *Algebraic methodology and software technology: 6th international conference, AMAST 97*, volume 1349 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

5. CoFI Task Group on Language Design. CASL – The CoFI Algebraic Specification Language – Summary. CoFI Document: CASL/Summary. WWW¹², FTP¹³, September 1997.
6. CoFI Task Group on Semantics. CASL – The CoFI Algebraic Specification Language (version 0.97) – Semantics. CoFI Note: S-6. WWW¹⁴, FTP¹⁵, July 1997.
7. R. Diaconescu and K. Futatsugi. Logical semantics of CafeOBJ. Technical report IS-RR-96-0024S, JAIST, 1996.
8. J. A. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1:49–67, 1991.
9. J. A. Goguen and R. M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39:95–146, 1992. Predecessor in: LNCS 164, 221–256, 1984.
10. J. A. Goguen and J. Meseguer. Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.
11. J. A. Goguen and T. Winkler. Introducing OBJ3. Research report SRI-CSL-88-9, SRI International, 1988.
12. A. Haxthausen and F. Nickl. Pushouts of order-sorted algebraic specifications. In *Proceedings of AMAST'96*, volume 1101 of *Lecture Notes in Computer Science*, pages 132–?? Springer-Verlag, 1996.
13. T. Mossakowski. Equivalences among various logical frameworks of partial algebras. In H. K. Büning, editor, *Computer Science Logic. 9th Workshop, CSL'95. Paderborn, Germany, September 1995, Selected Papers*, volume 1092 of *Lecture Notes in Computer Science*, pages 403–433. Springer Verlag, 1996.
14. T. Mossakowski, Kolyang, and B. Krieg-Brückner. Static semantic analysis of CASL. 12th Workshop on Algebraic Development Techniques, Tarquinia. This volume, 1997.
15. P. D. Mosses. CoFI: The common framework initiative for algebraic specification and development. In M. Bidoit and M. Dauchet, editors, *Theory and Practice of Software Development*, LNCS 1214, pages 115– 137. Springer Verlag, 1997.
16. O. Owe and O.-J. Dahl. Generator induction in order sorted algebras. *Formal Aspects of Computing*, 3:2–20, 1991.

¹² <http://www.brics.dk/Projects/CoFI/Documents/CASL/Summary/>

¹³ <ftp://ftp.brics.dk/Projects/CoFI/Documents/CASL/Summary/>

¹⁴ <http://www.brics.dk/Projects/CoFI/Notes/S-6/>

¹⁵ <ftp://ftp.brics.dk/Projects/CoFI/Notes/S-6/>