Partial Pushout Semantics of Generics in GDOL

Till Mossakowski[a], Bernd Krieg-Brückner[b]

[a] *Institute for Intelligent Cooperating Systems, Faculty of Computer Science, Otto-von-Guericke-Universität Magdeburg, Germany*
[b] *German Research Center for Artificial Intelligence (DFKI), CPS, BAALL, and Universität Bremen, FB3 Mathematik und Informatik, Bremen, Germany*

Abstract

We combine CASL's pushout-style generic specification with DOL's filtering, the latter being a syntactic removal of parts of a specification. The challenge is that now the body of a generic specification can remove parts of the formal parameter. This cannot be handled with usual pushout semantics, but calls for a semantics of "match, delete, glue in" as used in the theory of graph grammars. We hence employ Heindel's theory of MipMap categories as a basis for the use of pushouts in categories of *partial* maps. We introduce a notion of MipMap institution that can serve as a semantic background for a partial pushout semantics of generics with filtering.

Keywords: generic specification; single-pushout transformation; institution; category of partial maps

1. Introduction

1.1. *Generic Specifications in CASL and filtering in DOL*

CASL, the *Common Algebraic Specification Language* [1, 20, 3], is a complex general-purpose specification language based on first-order logic with induction. CASL also provides powerful structuring constructs, including extensions, unions, translations and generic specifications. CASL has been extended to the *Distributed Ontology, Modeling and Specification Language*, DOL, an OMG standard, [21, 18, 17]. DOL provides explicit language constructs for selecting the logic, such that CASL's structuring constructs can be used for a variety of logics. Moreover, DOL provides further structuring constructs like filtering, which expresses the syntactic removal of parts of a specification. However, generic specifications are not part of DOL.

Generic Specifications in CASL are the basis for the semantics in GDOL described in this paper, which extends DOL with generics. They have a predecessor in SPECTRAL, a compact specification language for structuring specifications and programs in-the-large and in-the-small, which was designed by Don Sannella and Bernd Krieg-Brückner, cf. [15]. General higher-order functions yielding specifications were the basis for supporting the specification development process and to enhance re-usability. While this intention was kept in

CASL, higher-order functions on specifications were deemed not to be necessary; now, more than 25 years later, this assessment might be reconsidered eventually in GDOL.

However, specifications as parameters were retained in CASL and provide an extremely powerful and compact way for defining applicability constraints in applications (see below). Don was also instrumental in defining the semantics of CASL, together with the first author among others, cf. [2].

In this paper, we will show how CASL generics can be combined with DOL filtering in a semantically meaningful way, leading to the language GDOL.

1.2. Specification of containers in CASL

Consider the following specification of natural numbers and generic specification of lists with a length operation in CASL:

```
spec Nat =
  free type Nat ::= 0 | suc(Nat)
end
spec List[Nat then sort Elem] =
  free type List[Elem] ::=  [] | __::__(Elem; List[Elem])
  op length : List[Elem] -> Nat
  forall x : Elem; l : List[Elem]
  . length([]) = 0
  . length(x::l) = suc(length(l))
end
```

The specification `Nat` specifies the natural numbers as a free datatype (=term algebra); details of the syntax will be explained below. The parameter of the generic specification `List` is the specification `Nat` extended ("**then**") by a specification consisting just of the sort `Elem`. Lists are specified as a free datatype with two constructors (empty list and prepending an element to a list). Double underscores declare infix notation, hence `__::__` is written between its arguments.

An instantiation of the generic specification of lists with Booleans (obtaining lists of Booleans) looks as follows:

```
spec Boolean =
  free type Boolean ::= True | False
end

spec List_of_Boolean =
  List[Nat then Boolean fit sort Elem |-> sort Boolean]
end
```

Note that `Nat` implicitly is mapped identically here (it strictly speaking should not be instaniated, but is a fixed specification called "import" in CASL).

If we want to remove the dependency on natural numbers (for example, because we do not need them and want to speed up theorem proving), we can use DOL filtering:

```
spec List_filtered[sort Elem] =
  List[Nat then sort Elem]
  reject sort Nat
end
```

The result is the removal of sort `Nat` and all operations and axioms mentioning `Nat`. Note that CASL hiding has a different effect: here, `Nat` is only hidden from the "export interface", but not removed. In particular, for theorem proving purposes, `Nat` will be uncovered.

Consider now the specification of finite sets over some element sort in CASL:

```
spec Set [Nat then sort Elem] =
    generated type Set ::= empty | insert(Elem; Set)
    pred __is_in__ : Elem * Set
    forall e, e':Elem; S, S':Set
    . not (e is_in empty)
    . (e is_in insert (e', S)) <=> e = e' \/ (e is_in S)
    . S = S' <=> forall x:Elem . (x is_in S) <=> (x is_in S')
                                                    %(equal_sets)%
end
```

Again, we can use DOL filtering for removing the dependency on natural numbers:

```
spec Set_filtered[sort Elem] =
  Set[Nat then sort Elem]
  reject sort Nat
end
```

Consider now the generic removal from `Nat` from containers like `List` and `Set`:

```
spec Container_filtered[Nat then sort Container] =
  sort Container
  reject sort Nat
end
```

This generic specification can be instantiated with lists and sets as defined above, but also with other containers like bags or multisets.

1.3. Generic Ontology Design Patterns

Another motivation for this paper stems from research on *Generic Ontology Design Patterns*, GODPs, [13, 14], with a focus on the *methodological perspective* of providing *safe ontology development.* GODPs abstract away from application domains to a generic methodological level, and are then instantiated to particular domains as part of an ontology engineering process. A repository of general GODPs can be used to compile specific application-oriented patterns.

Since a GODP abstracts from a particular development fragment, it embodies an *ontology development operation*, to be re-used in a variety of contexts.

Parameters to a GODP may be classes, properties, individuals, or whole ontologies. *Semantic pre-conditions* may be stated by axioms in parameter ontologies. *Consistency* is ensured by proving each ontology argument to comply with its parameter.

A GODP is *safe* insofar as it can be separately defined and proved correct. *Changes are confined* to the effects of GODP instantiation, checking arguments for violation of semantic constraints; no other part of the host ontology is accidentally affected.

The examples in [14] only treat generic specifications without filtering, i.e. where the body extends the formal parameter, as allowed in CASL. However, as an example for the general case of "repairing" some axiom to additionally include an extension, consider a pattern for subclasses Z1 and Z2 of X with a disjoint union axiom for the subclasses (we use the Web Ontology Language OWL with Manchester syntax here, see the next section for some details):

```
ontology DisjointnessExtension
 [Class: X   Class: Z1 SubClassOf: X   Class: Z2 SubClassOf: X]
 [Class: Y] =
 Class: Y SubClassOf: X
 DisjointClasses: Y,Z1,Z2
 reject DisjointClasses: Z1,Z2
end
```

If we now want to add another subclass Y, we have to "repair" the disjoint union axiom to include Y by first rejecting the old axiom and then introducing the extended one.

2. Institutions

The notion of institution [8] formalises the informal notion of logical system and is used in the semantics of structuring and generic specification in both CASL and DOL.

Definition 1. An *institution* is a quadruple $I = (\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ consisting of the following:

- a category $\mathbf{Sign}$ of *signatures*,
- a functor $\mathbf{Sen}\colon \mathbf{Sign} \longrightarrow \mathbb{S}et$[1] giving, for each signature Σ, the set of *sentences* $\mathbf{Sen}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *sentence translation map* $\mathbf{Sen}(\sigma)\colon \mathbf{Sen}(\Sigma) \longrightarrow \mathbf{Sen}(\Sigma')$, where often $\mathbf{Sen}(\sigma)(\varphi)$ is written as $\sigma(\varphi)$,
- a functor $\mathbf{Mod}\colon \mathbf{Sign}^{op} \longrightarrow \mathbb{C}at$[2] giving, for each signature Σ, the category of *models* $\mathbf{Mod}(\Sigma)$, and for each signature morphism $\sigma\colon \Sigma \longrightarrow \Sigma'$, the *reduct functor* $\mathbf{Mod}(\sigma)\colon \mathbf{Mod}(\Sigma') \longrightarrow \mathbf{Mod}(\Sigma)$, where often $\mathbf{Mod}(\sigma)(M')$

[1] $\mathbb{S}et$ is the category having all small sets as objects and functions as arrows.
[2] $\mathbb{C}at$ is the category of categories and functors. Strictly speaking, $\mathbb{C}at$ is not a category but only a so-called quasicategory, which is a category that lives in a higher set-theoretic universe.

is written as $M'{\restriction}_\sigma$, and $M'{\restriction}_\sigma$ is called the *σ-reduct* of M', while M' is called a *σ-expansion* of $M'{\restriction}_\sigma$,

- a satisfaction relation $\models_\Sigma \subseteq |\mathbf{Mod}(\Sigma)| \times \mathbf{Sen}(\Sigma)$ for each $\Sigma \in |\mathbf{Sign}|$,

such that for each $\sigma\colon \Sigma \longrightarrow \Sigma'$ in **Sign** the following *satisfaction condition* holds:

$$(\star) \qquad M' \models_{\Sigma'} \sigma(\varphi) \text{ iff } M'{\restriction}_\sigma \models_\Sigma \varphi$$

for each $M' \in |\mathbf{Mod}(\Sigma')|$ and $\varphi \in \mathbf{Sen}(\Sigma)$, expressing that truth is invariant under change of notation and context. □

Example 1. *Many-sorted first-order logic with equality* $\mathbf{MSFOL}^=$. A *many-sorted signature* $\Sigma = (S, F, P)$ consists of a set S of *sorts*, a $S^*{\times}S$-indexed familiy F *function symbols*, where constants are treated as functions with no arguments, and an S^*-indexed family P of *predicate symbols*. We write $f : w \to s \in F$ for $f \in F_{w,s}$ (with $f : s$ for empty w), and $p : w \in P$ for $p \in P_w$.

Given signatures Σ and Σ', a *signature morphism* $\sigma\colon \Sigma \longrightarrow \Sigma'$ maps sorts, function symbols and predicate symbols in Σ to symbols of the same kind in Σ'. Profiles must be preserved, so for instance $f : w \to s$ in Σ maps to a function symbol in Σ' with profile $\sigma^*(w) \to \sigma(s)$, where σ^* is the extension of σ to finite strings of symbols. Identities and composition are defined in the obvious way, giving a category **Sign** of $\mathbf{MSFOL}^=$-signatures.

Given a finite string $w = s_1 \dots s_n$ and sets $M_{s_1}, \dots, M_{s_n}$, we write M_w for the Cartesian product $M_{s_1} \times \dots \times M_{s_n}$. Let $\Sigma = (S, F, P)$. A *many-sorted Σ-model* M consists of a non-empty *carrier set* M_s for each sort $s \in S$, a total function $(f_{w,s})_M : M_w \to M_s$ for each total function symbol $f : w \to s \in F$, and a predicate $(p_w)_M \subseteq M_w$ for each predicate symbol $p : w \in P$.

Model homomorphisms are defined in the standard way.

Concerning *reducts*, if $\sigma\colon \Sigma \longrightarrow \Sigma'$ is a signature morphism and M' is a Σ'-model, then $M'|_\sigma$ is a Σ-model with $(M'|_\sigma)_s := M'_{\sigma(s)}$ for $s \in S$ and analogously for $(f_{w,s})_{M'|_\sigma}$ and $(p_w)_{M'|_\sigma}$. Reducts of homomorphisms are defined similarly.

First-order sentences over a signature are defined in the usual way, using logical connectives and quantified variables. Sentence translation along a signature morphism means replacement of the translated symbols. Finally, satisfaction of a sentence in a model follows the usual Tarskian definition. For further details, see [8]. □

Example 2. *Many-sorted first-order logic with equality and sort generation constraints* $\mathbf{MSCFOL}^=$. $\mathbf{MSCFOL}^=$ extends $\mathbf{MSFOL}^=$ with *sort generation constraints*. Given a many-sorted signature $\Sigma = (S, F, P)$, a sort generation constraint is a pair (S_0, F_0), where $S_0 \subseteq S$ and $F_0 \subseteq F$.[3] A model M satisfies (S_0, F_0), if for each element a of a carrier of a sort in S_0, there is a term t in operations from F_0 and variables sorted outside S_0, such that t evaluates to a for some valuation of the variables. □

[3]Strictly speaking, sort generation constraints need a third component, a signature morphism, ensuring that they can safely be translated along signature morphisms.

Sort generation constraints correspond to induction principles and can be used for the specification of inductive datatypes. Consider the following specification of natural numbers:

```
spec Nat =
  free type Nat ::= 0 | suc(Nat)
end
```

This is CASL shorthand notation for the following presentation in **MSCFOL**$^=$:

```
spec Nat_expanded =
  sort Nat
  ops 0 : Nat; suc : Nat -> Nat
  forall X1 : Nat; Y1 : Nat
  . suc(X1) = suc(Y1) <=> X1 = Y1 %(ga_injective_suc)%
  . not 0 = suc(Y1) %(ga_disjoint_0_suc)%
  generated type Nat ::= 0 | suc(Nat) %(ga_generated_Nat)%
end
```

Here, the last axiom is the CASL notation for the sort generation constraint $(\{\mathtt{Nat}\}, \{\mathtt{0}, \mathtt{suc}\})$.

Example 3. *Description Logics.* Signatures of the description logic $\mathcal{ALC}$ consist of a set $\mathcal{A}$ of atomic concepts, a set $\mathcal{R}$ of roles and a set $\mathcal{I}$ of individual constants, while signature morphisms provide respective mappings. Models are single-sorted first-order structures that interpret concepts as unary and roles as binary predicate symbols. Sentences are subsumption relations $C_1 \sqsubseteq C_2$ between concepts, where concepts follow the grammar

$$C ::= \mathcal{A} \,|\, \top \,|\, \bot \,|\, C_1 \sqcup C_2 \,|\, C_1 \sqcap C_2 \,|\, \neg C \,|\, \forall R.C \,|\, \exists R.C$$

These kind of sentences are also called TBox sentences. Sentences can also be ABox sentences, which are membership assertions of individuals in concepts (written $a : C$ for $a \in \mathcal{I}$) or pairs of individuals in roles (written $R(a, b)$ for $a, b \in \mathcal{I}, R \in \mathcal{R}$). Sentence translation and reduct is defined similarly as in **MSFOL**$^=$. Satisfaction is the standard satisfaction of description logics. See [16] for a formalisation as an institution.

The logic $\mathcal{SROIQ}$ [11], which is the logical core of the Web Ontology Language $\mathcal{OWL}$-DL 2.0[4] extends $\mathcal{ALC}$ with the following constructs: (i) complex role boxes (denoted by $\mathcal{SR}$): these can contain: complex role inclusions such as $R \circ S \sqsubseteq S$ as well as simple role hierarchies such as $R \sqsubseteq S$, assertions for symmetric, transitive, reflexive, asymmetric and disjoint roles (called RBox sentences), as well as the construct $\exists R.\mathsf{Self}$ (collecting the set of 'R-reflexive points'); (ii) nominals (denoted by $\mathcal{O}$); (iii) inverse roles (denoted by $\mathcal{I}$); qualified and unqualified number restrictions ($\mathcal{Q}$). For details on the rather complex grammatical restrictions for $\mathcal{SROIQ}$ (e.g. regular role inclusions, simple roles)

[4]See also `http://www.w3.org/TR/owl2-overview/`

compare [11], and see the example given below. $\mathcal{SROIQ}$ can be straightforwardly rendered as an institutions following the previous examples, but compare also [16]. $\square$

Within an arbitrary but fixed institution, we can easily define the usual notion of *logical consequence* or *semantical entailment*: Given a set of Σ-sentences Γ and a Σ-sentence φ, we say

$$\Gamma \models_\Sigma \varphi \text{ (φ follows from Γ)}$$

iff for all Σ-models M, we have

$$M \models_\Sigma \Gamma \text{ implies } M \models_\Sigma \varphi.$$

Here, $M \models_\Sigma \Gamma$ means that $M \models_\Sigma \psi$ for each $\psi \in \Gamma$.

Given a set of Σ-sentences Γ, we write $\Gamma^\bullet$ for the set $\{\varphi \in \mathbf{Sen}(\Sigma) \mid \Gamma \models \varphi\}$ of logical consequences of Γ. $\Gamma_1, \Gamma_2 \subseteq \mathbf{Sen}(\Sigma)$ are *logically equivalent*, written $\Gamma_1 \models\!\mid \Gamma_2$, if $\Gamma_1 \models \Gamma_2$ and $\Gamma_2 \models \Gamma_1$ (or equivalently, $\Gamma_1^\bullet = \Gamma_2^\bullet$).

In an arbitrary institution I, a *presentation* is a pair $T = \langle \Sigma, \Gamma \rangle$, where $\Sigma \in \mathbf{Sign}$ and $\Gamma \subseteq \mathbf{Sen}(\Sigma)$. We denote Σ with T_Σ and Γ with T_Γ. *Presentation morphisms* $\sigma: \langle \Sigma, \Gamma \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ are those signature morphisms $\sigma: \Sigma \longrightarrow \Sigma'$ for which $\Gamma' \models_{\Sigma'} \sigma(\Gamma)$, or, in other words $\sigma(\Gamma) \subseteq \Gamma'^\bullet$. By inheriting composition and identities from $\mathbf{Sign}$, we obtain a category $\mathbf{Pres}$ of presentations.

A presentation morphism $\sigma: \langle \Sigma, \Gamma \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ is *conservative*, if $\Gamma \models \sigma^{-1}(\Gamma')$.

We will also freely use other standard logical terminology when working within an arbitrary but fixed institution. Moreover, in such an arbitrary institution, we can use *filtering*.

2.1. Filtering in DOL

We now introduce filtering in DOL [17] in a simplified form. For filtering to be well-defined, we need to assume some inclusion system [4] on the signature category, such that set-theoretic notions on signatures make sense.

Filtering takes a presentation $\langle \Sigma, \Gamma \rangle$ and removes some parts. A filtering

$$\langle \Sigma, \Gamma \rangle \textbf{ select } \langle \Sigma_0, \Gamma_0 \rangle$$

(with $\Sigma_0 \subseteq \Sigma$ and $\Gamma_0 \subseteq \Gamma$) selects those sentences from $\langle \Sigma, \Gamma \rangle$ that have signature Σ_0, plus those in Γ_0. We can also write

$$\langle \Sigma, \Gamma \rangle \textbf{ reject } \langle \Sigma_0, \Gamma_0 \rangle$$

where Σ_0 is the set of symbols and Γ_0 the set of axioms to be hidden. For example, we can select all axioms of the GALEN anatomy ontology[5] involving Drugs, Joints, or Bodyparts by:

[5]We assume that GALEN is available as an OWL ontology.

```
logic OWL
ontology myGalen =
   <http://example.org/GALEN/galen.owl>
 select Drugs, Joints, Bodyparts
end
```

The semantics of $\langle \Sigma, \Gamma \rangle$ **select** $\langle \Sigma_0, \Gamma_0 \rangle$ is given by the presentation $\langle \Sigma', \Gamma' \rangle$, where

- Σ' is the smallest signature with $\Sigma_0 \subseteq \Sigma' \subseteq \Sigma$ and $\Gamma_0 \subseteq \mathbf{Sen}(\Sigma')$[6] and
- $\Gamma' = (\Gamma \cap \mathbf{Sen}(\Sigma')) \cup \Gamma_0$.

The semantics of $\langle \Sigma, \Gamma \rangle$ **reject** $\langle \Sigma_0, \Gamma_0 \rangle$ is given by the presentation $\langle \Sigma', \Gamma' \rangle$, where

- $\Sigma' = \Sigma \setminus \Sigma_0$ and
- $\Gamma' = (\Gamma \cap \mathbf{Sen}(\Sigma')) \setminus \Gamma_0$.

3. Semantics for generics with filtering

We now investigate how a semantics of generics with filtering in GDOL that builds on the current semantics generic in CASL and filtering in DOL could look.

3.1. Double-pushout semantics

The well-known pushout-style semantics [7, 20] formalises a generic specification as a morphism $PAR \to BODY$. The application of such a generic specification to an argument specification ARG requires a fitting morphism $PAR \to ARG$. The resulting specification is then obtained as the following pushout:

$$\begin{tikzcd}
PAR \arrow[r] \arrow[d] & BODY \arrow[d] \\
ARG \arrow[r] & RESULT \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

However, this prevents symbols in the parameter from being deleted in the body — the paramater is always included in the body. The possibility of deleting symbols from the parameter requires a more general situation like this:

$$\begin{tikzcd}
PAR \arrow[d] & INTERFACE \arrow[l] \arrow[r] \arrow[d] & BODY \arrow[d] \\
ARG & ARG\_INTERFACE \arrow[l] \arrow[r] & RESULT
\end{tikzcd}$$

[6]If this smallest signature does not exist, the semantics is undefined.

This resembles the double-pushout approach used in algebraic graph transformation. Indeed, graph transformation rules correspond to embeddings of the parameter into the body of a generic specification, and matches between the left hand-side of a rule correspond to fitting morphisms mapping the parameter to the argument instantiating a generic specification.

The proper abstract framework for the double-pushout approach to algebraic graph transformations is that of (weak) adhesive HLR categories [5]. A crucial result for weak adhesive HLR categories is the following:

Theorem 1 (Uniqueness of pushout complements [5]). *Given* $k : INTERFACE \to PAR \in \mathcal{M}$ *and* $s : PAR \to ARG$*, then there is, up to isomorphism, at most one* $ARG\_INTERFACE$ *with* $l : INTERFACE \to ARG\_INTERFACE$ *and* $u : ARG\_INTERFACE \to ARG$ *such that the following is a pushout.*

$$\begin{tikzcd}
PAR \arrow[d, "s"] & INTERFACE \arrow[l, "k"'] \arrow[d, "l"] \\
ARG \arrow[ur, phantom, "\urcorner", very near start] & ARG\_INTERFACE \arrow[l, "u"]
\end{tikzcd}$$

The property that such a pushout exists is called the *glueing condition.* In case that the glueing condition holds, the application of the generic specification

$$\begin{tikzcd}
PAR & INTERFACE \arrow[l] \arrow[r] & BODY
\end{tikzcd}$$

to an argument ARG is then given by

$$\begin{tikzcd}
PAR \arrow[d] & INTERFACE \arrow[l] \arrow[r] \arrow[d] & BODY \arrow[d] \\
ARG \arrow[ur, phantom, "\urcorner", very near start] & ARG\_INTERFACE \arrow[l] \arrow[r] & RESULT \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

A counterexample (formulated in CASL [20, 3]) to the glueing condition is the following:

```
spec DELETE_SORT[sorts s,t] = reject t then sort u end
spec DELETE_SORT_INST = DELETE_SORT[sorts s,t; ops f:s->s; g:s->t]
```

leading to the following diagram:

$$\begin{tikzcd}
\textbf{sorts}\ \texttt{s,t} \arrow[d] & \textbf{sort}\ \texttt{s} \arrow[l] \\
\textbf{sorts}\ \texttt{s,t;}\ \textbf{ops}\ \texttt{f:s->s; g:s->t} &
\end{tikzcd}$$

This cannot be complemented to a pushout, because the operation `op g:s->t` is “dangling”: the sort `t` is missing in the interface. Still, we would like such a generic specifications to be well-formed, with the following result, where the dangling operation is just deleted:

$$\begin{tikzcd}
\textbf{sorts}\ \texttt{s,t} \arrow[d] & \textbf{sort}\ \texttt{s} \arrow[l] \arrow[r] \arrow[d] & \textbf{sort}\ \texttt{s,u} \arrow[d] \\
\textbf{sorts}\ \texttt{s,t;}\ \textbf{ops}\ \texttt{f:s->s; g:s->t} & \textbf{sort}\ \texttt{s} \arrow[l] \arrow[r] & \textbf{sort}\ \texttt{s,u;}\ \textbf{op}\ \texttt{f:s->s}
\end{tikzcd}$$

3.2. Single-pushout semantics with partial maps

This bevaviour can be achieved using the single-pushout (SPO) approach to graph transformation [6], where pushouts in a category of *partial* maps are used:

$$\begin{tikzcd}
PAR \arrow[r, "\circ" marking] \arrow[d] & BODY \arrow[d] \\
ARG \arrow[r, "\circ" marking] \arrow[ur, phantom, "\ulcorner" near end] & RESULT
\end{tikzcd}$$

Here, a partial map $PAR \xrightarrow{\circ} BODY$ is represented by a span

$$PAR \longleftarrow INTERFACE \longrightarrow BODY$$

where the left arrow is a suitable monomorphism representing the domain of the partial map. The existence of pushouts of such partial maps is guaranteed in MipMap categories [9]. We hence recall this notion. It depends on a suitably chosen class $\mathcal{M}$ of monomorphisms representing domains of the partial maps. Suitability of such a choice is defined as follows:

Definition 2 (Admissible class of monos [22]). A class $\mathcal{M}$ of monos in a category $\mathbf{C}$ is *admissible*, if

1. $\mathbf{C}$ has pullbacks along $\mathcal{M}$-morphisms,
2. $\mathcal{M}$ is stable under pullback,
3. $\mathcal{M}$ contains all identities, and
4. $\mathcal{M}$ is closed under composition.

In the sequel, we assume that an arbitrary but fixed category $\mathbf{C}$ with an admissible class $\mathcal{M}$ is given. For an object A in $\mathbf{C}$, we can then define the lattice $(Sub_{\mathcal{M}}A, \sqsubseteq)$ of subobjects of A consisting of equivalence classes $[m]$ of morphisms $m : M \hookrightarrow A \in \mathcal{M}$ taken up to isomorphism in the comma category $(\mathbf{C}, A)$. The ordering is defined by $m \sqsubseteq n$ if there exists some i with $m = i; n$.

Partial maps with domains from $\mathcal{M}$ are defined as follows:

Definition 3 (Category of partial maps [22]). The category $\mathbf{C}_{*\mathcal{M}}$ of $\mathcal{M}$-*partial maps* has the same objects as $\mathbf{C}$. Morphisms from $[(m, X, f)] : A \to B$ are spans $A \xleftarrow{m} X \xrightarrow{f} B$ in $\mathbf{C}$ with $m \in \mathcal{M}$, taken up to isomorphism of spans. Identities in $\mathbf{C}_{*\mathcal{M}}$ are $[(id_A, A, id_A)]$ and composition is defined by pulling back. A partial map $[(m, X, f)]$ is called *total* if m is an isomorphism. The embedding functor $\Gamma : \mathbf{C} \to \mathbf{C}_{*\mathcal{M}}$ is the identity on objects and maps a morphism $f : A \to B$ to the total map $[(id_A, A, f)] : A \to B$.

Definition 4 (Inverse image [22]). Given a morphism $f : A \to B \in \mathbf{C}_{*\mathcal{M}}$, we can take inverse images against f using pullbacks. The inverse image function $f^{-1} : Sub_{\mathcal{M}}B \to Sub_{\mathcal{M}}A$ takes $[m] \in Sub_{\mathcal{M}}B$ to $[m'] \in Sub_{\mathcal{M}}A$ given by a pullback

$$\begin{tikzcd}
M' \arrow[r] \arrow[d, tail, "m'"] \arrow[dr, phantom, "\lrcorner", very near start] & M \arrow[d, tail, "m"] \\
A \arrow[r, "f"] & B
\end{tikzcd}$$

Definition 5 (Upper adjoint to inverse image [9]). Given a morphism $f : A \to B \in \mathbf{C}_{*\mathcal{M}}$, a monotone function $\mathcal{U} : Sub_{\mathcal{M}}A \to Sub_{\mathcal{M}}B$ is an *upper adjoint* to f^{-1} if for all $n \in Sub_{\mathcal{M}}B$ and all $m \in Sub_{\mathcal{M}}A$, we have

$$f^{-1}(n) \sqsubseteq m \text{ iff } n \sqsubseteq \mathcal{U}(m)$$

If such an upper adjoint exists, it is denoted by $\forall_f$.

Definition 6 (Hereditary pushout [12]). A pushout in $\mathbf{C}$ is *hereditary* if Γ maps it to a pushout in $\mathbf{C}_{*\mathcal{M}}$.

This terminology now allows for a characterisation of the existence of pushouts of partial maps:

Theorem 2 (Existence of pushouts of partial maps [9]). *Given a category* $\mathbf{C}$ *with cocones of spans and an admissible class of monos* $\mathcal{M}$*, the category of partial maps* $\mathbf{C}_{*\mathcal{M}}$ *has pushouts if and only if* $\mathbf{C}$ *has hereditary pushouts and upper adjoints of inverse images.*

This theorem justifies the following terminology:

Definition 7 (MipMap category [9]). A category $\mathbf{C}$ with an admissible class of monos $\mathcal{M}$ is called a *MipMap category*, if $\mathbf{C}$ has hereditary pushouts and upper adjoints of inverse images.

An easy way to show that a category is MipMap is to show that it is a topos:

Theorem 3 ([9]). *Each topos is MipMap.*

In order to use the notion of MipMap category for institutions, we need a slight additional condition basically ensuring that inverse images of symbol sets lead to inverse images of sentence sets.

Definition 8 (MipMap institution). An institution is a *MipMap institution* if its signature category is MipMap, and moreover the sentence functor maps pullbacks along $\mathcal{M}$-morphisms to pullbacks in $\mathbb{S}et$.

Example 4. $\mathcal{OWL}$ is a MipMap institution, because its signature category is $\mathbb{S}et^3$, which is a (presheaf) topos, and hence MipMap by Thm. 3. $\mathcal{M}$ is the class of all monomorphisms (i.e. the component-wise injections.) Preservation of pullbacks along morphisms in $\mathcal{M}$ is straightforward to show. $\square$

Example 5. $\mathbf{MSFOL}^=$ is a MipMap institution. $\mathcal{M}$ is the class of all monomorphisms. Again, preservation of pullbacks along morphisms in $\mathcal{M}$ is straightforward to show. We show that the $\mathbf{MSFOL}^=$ signature category is a topos; then the MipMap property follows by Thm. 3.

We first need to show the existence of finite limits. The terminal $\mathbf{MSFOL}^=$ signature 1 is given by one sort, one function symbol and one predicate symbol of each arity. Concerning products, given $\Sigma_1 = (S_1, F_1, P_1)$ and $\Sigma_2 = (S_2, F_2, P_2)$, the product $\Sigma_1 \times \Sigma_2 = (S, F, P)$ has sorts $S = S_1 \times S_2$. $F_{(s_1,t_1)\dots(s_n,t_n)\to(s,t)} = (F_1)_{s_1\dots s_n\to s} \times (F_2)_{t_1\dots t_n\to t}$, that is, if $f_1 : s_1 \dots s_n \to s \in F_1$ and $f_2 : t_1 \dots t_n \to t \in F_2$, then $(f_1, f_2) : (s_1, t_1) \dots (s_n, t_n) \to (s, t) \in F$. Likewise, $P_{(s_1,t_1)\dots(s_n,t_n)} = (P_1)_{s_1\dots s_n} \times (P_2)_{t_1\dots t_n}$. Concerning equalisers, given $\Sigma_1 = (S_1, F_1, P_1)$ and $\Sigma_2 = (S_2, F_2, P_2)$ and $\sigma_1, \sigma_2 : \Sigma_1 \to \Sigma_2$, the equaliser

$$\begin{tikzcd}(S,F,P) = \Sigma \arrow[r, tail, "\sigma"] & \Sigma_1 \arrow[r, shift left, "\sigma_1"] \arrow[r, shift right, "\sigma_2"'] & \Sigma_2\end{tikzcd}$$

is given by $S = \{s \in S_1 \mid \sigma_1(s) = \sigma_2(s)\}$, $F = \{f : w \to s \in F_1 \mid \sigma_1(f : w \to s) = \sigma_2(f : w \to s)\}$ and $P = \{p : w \in P_1 \mid \sigma_1(p : w) = \sigma_2(p : w)\}$. σ is the inclusion.

Next we come to exponentials. Given $\Sigma_1 = (S_1, F_1, P_1)$ and $\Sigma_2 = (S_2, F_2, P_2)$, $\Sigma_1^{\Sigma_2} = (S, F, P)$ has sorts $S = S_1^{S_2}$. Moreover, $(f, s_1, \dots, s_n, s) : t_1 \dots t_n \to t \in F$ iff $t_1 = \dots = t_n = t$, $f \in \left((F_1)_{t(s_1)\dots t(s_n)\to t(s)}\right)^{(F_2)_{s_1\dots s_n\to s}}$ and $s_1 \dots s_n, s \in S_2$. For predicate symbols, $(p, s_1, \dots, s_n) : t_1 \dots t_n \in P$ iff either $n > 0$, $t_1 = \dots = t_n$, $p \in \left((P_1)_{t_1(s_1)\dots t_1(s_n)}\right)^{(P_2)_{s_1\dots s_n}}$ and $s_1 \dots s_n \in S_2$, or $n = 0$ and $p \in ((P_1)_\varepsilon)^{(P_2)_\varepsilon}$, where ε is the empty word. The evaluation morphism $eval : \Sigma_2 \times \Sigma_1^{\Sigma_2} \to \Sigma_1$ is like that in $\mathbb{S}et$ for sorts, i.e. a pair $(s \in S_2, t \in S_1^{S_2})$ is mapped to $t(s)$. Concerning operation symbols, a pair $(f : s_1 \dots s_n \to s \in \Sigma_2, (h, s_1, \dots, s_n, s) : t \dots t \to t \in \Sigma_1^{\Sigma_2})$ is mapped to $h(f) : t(s_1) \dots t(s_n) \to t(s) \in \Sigma_1$. Likewise, a predicate symbol $(p : s_1 \dots s_n \in \Sigma_2, (h, s_1, \dots, s_n) : t \dots t \in \Sigma_1^{\Sigma_2})$ is mapped to $h(p) : t(s_1) \dots t(s_n) \in \Sigma_1$. It is straightforward to show the universal property, which means that we have shown cartesian closedness.

Finally, the subobject classifier $\Omega = (S, F, P)$ is given by sorts $S = \{\top, \bot\}$. Operation symbols are $\top : \top \dots \top \to \top$ (arbitrarily many arguments of sort $\top$) and $\bot : s_1 \dots s_n \to s$ where $s_1, \dots, s_n, s \in \{\top, \bot\}$. Similarly for predicate symbols. The signature morphism $\top : 1 \to \Omega$ maps to the sort $\top$, operation symbols $\top : \top \dots \top \to \top$ and predicate symbols $\top : \top \dots \top$. Let us now show the subobject classifier property. Given a signature monomorphism $\Sigma_0 \xrightarrow{\sigma} \Sigma$ with $\Sigma_0 = (S_0, F_0, P_0)$, let $\chi_\sigma : \Sigma \to \Omega$ act on sorts as

$$\chi_\sigma(s) = \begin{cases} \top, & \text{if } s \in \sigma(S_0) \\ \bot, & \text{otherwise} \end{cases},$$

on operation symbols as

$$\chi_\sigma(f : s_1 \dots s_n \to s) = \begin{cases} \top{:}\chi_\sigma(s_1)\dots\chi_\sigma(s_n) \to \chi_\sigma(s), & \text{if } f : s_1 \dots s_n \to s \in \sigma(F_0) \\ \bot{:}\chi_\sigma(s_1)\dots\chi_\sigma(s_n) \to \chi_\sigma(s), & \text{otherwise} \end{cases}$$

and similarly on predicate symbols. Then it is straightforward to show that

$$\begin{tikzcd}
\Sigma_0 \arrow[r, "!"] \arrow[d, tail, "\sigma"] & 1 \arrow[d, tail, "\top"] \\
\Sigma \arrow[r, "\chi_\sigma"] & \Omega
\end{tikzcd}$$

is a pullback. Moreover, χ_σ is unique with this property. □

3.3. Semantics of generics with filtering

Our central result provides the semantics of generics with filtering:

Theorem 4. *In a MipMap institution with admissible class of monos $\mathcal{M}^{\mathbf{Sign}}$, the category of presentations is MipMap, if its admissible class of monos $\mathcal{M}^{\mathbf{Pres}}$ is taken to be all conservative presentation morphisms with underlying signature morphism in $\mathcal{M}^{\mathbf{Sign}}$.*

This means that in a MipMap institution, for each span of presentations representing a generic specification and each fitting map into an argument presentation

$$\begin{tikzcd}
PAR \arrow[d] & INTERFACE \arrow[l] \arrow[r] & BODY \\
ARG & &
\end{tikzcd}$$

the instantiating pushout exists.

In order to prove theorem 4, we first need to cite the following characterisation:

Theorem 5 (Characterisation of hereditary pushouts [10, 12]). *A pushout in* $\mathbf{C}$

$$\begin{tikzcd}
& A \arrow[r, "g"] \arrow[dl, "f"'] & C \arrow[dl, "k"] \\
B \arrow[r, "h"] & D \arrow[ul, phantom, "\ulcorner"] &
\end{tikzcd}$$

is hereditary if and only if for every completion to a cube

$$\begin{tikzcd}
& A' \arrow[rr, "g'"] \arrow[dl, "f'"'] \arrow[dd, tail, "a"] & & C' \arrow[dl, "k'"] \arrow[dd, tail, "c"] \\
B' \arrow[rr, "h'"] \arrow[dd, tail, "b"] & & D' \arrow[dd, "d"] & \\
& A \arrow[rr, "g"] \arrow[dl, "f"] & & C \arrow[dl, "k"] \\
B \arrow[rr, "h"] & & D &
\end{tikzcd}$$

with $b, c \in \mathcal{M}$ and the back faces pullbacks, the top face is a pushout iff the front faces are pullbacks and $d \in \mathcal{M}$.

Proof (of Theorem 4). In the following, we denote presentations with $A, B, C, \ldots$ and assume that $A = (\Sigma_A, \Gamma_A)$ etc. It is well-known [8] that colimits, hence in particular pushouts, lift from the signature category to the category of presentations. For constructing a pushout

$$\begin{tikzcd}
A \arrow[r, "g"] \arrow[d, "f"] & C \arrow[d, "f'"] \\
B \arrow[r, "g'"] \arrow[ur, phantom, "\ulcorner"] & D
\end{tikzcd}$$

we can take $\Gamma_D = f'(\Gamma_C) \cup g'(\Gamma_B)$ (actually, any logically equivalent presentation will do).

Concerning pullbacks of morphisms $f : A \to B$ along morphisms $m \in \mathcal{M}^{\mathbf{Pres}}$,

$$\begin{tikzcd}
& A \arrow[r, "f"] \arrow[dr, phantom, "\lrcorner"] & B \\
& N \arrow[u, tail, "n"] \arrow[r, "g"] & M \arrow[u, tail, "m"'] \\
N' \arrow[uur, bend left, "n'"] \arrow[ur, dotted, "h"] \arrow[urr, bend right, "f'"'] & &
\end{tikzcd}$$

we construct them in **Sign** and set

$$\Gamma_N = n^{-1}(\Gamma_A^{\bullet}),$$

ensuring that $n \in \mathcal{M}^{\mathbf{Pres}}$ and hence $\mathcal{M}^{\mathbf{Pres}}$ is stable under pullbacks. For well-definedness of the pullback square, we need to show that g is a presentation morphism, i.e. $\Gamma_N \subseteq g^{-1}(\Gamma_M^{\bullet})$. This is the same as $n^{-1}(\Gamma_A^{\bullet}) \subseteq g^{-1}(m^{-1}(\Gamma_B^{\bullet}))$.

The latter term is $n^{-1}(f^{-1}(\Gamma_B^\bullet))$, and hence the inclusion follows from f being a presentation morphism. In order to show the pullback property, consider a cocone (N', n', f') as shown above. By the pullback property in **Sign**, there exists a unique mediating signature morphism $h : n' \to N$ making the triangles commute. We need to show that it is a presentation morphism. Indeed, from $n'(\Gamma_{N'}) \subseteq \Gamma_A^\bullet$, we obtain $n(h(\Gamma_{N'})) \subseteq \Gamma_A^\bullet$ and thus $h(\Gamma_{N'}) \subseteq n^{-1}(\Gamma_A^\bullet) = \Gamma_N$. Altogether, we have shown that $\mathcal{M}^{\mathbf{Pres}}$ is admissible.

Next, we need to show that pushouts in **Pres** are hereditary, assuming that they are so in **Sign**. We use Thm. 5 and its notation. Assume that the bottom face is a pushout (i.e. $\Gamma_D \models\!\mid h(\Gamma_B) \cup k(\Gamma_C))$ and the back faces are pullbacks (in **Pres**), and $b, c \in \mathcal{M}^{\mathbf{Pres}}$ (hence also $a \in \mathcal{M}^{\mathbf{Pres}}$).

For the "only if" direction, assume that also the top face is a pushout. Since pushouts in **Sign** are hereditary, this means that the front faces are pullbacks in **Sign** and $d \in \mathcal{M}^{\mathbf{Sign}}$. Now let us show that $d \in \mathcal{M}^{\mathbf{Pres}}$. Suppose that $\varphi_{D'} \in d^{-1}(\Gamma_D^\bullet)$, i.e. $d(\varphi_{D'}) \in \Gamma_D^\bullet$. W.l.o.g. let us assume that $d(\varphi_{D'}) \in h(\Gamma_B^\bullet)$ (the case $d(\varphi) \in k(\Gamma_C^\bullet)$ is treated similarly). This means that there is $\varphi_B \in \Gamma_B^\bullet$ with $h(\varphi_B) = d(\varphi_{D'})$. Since **Sen** preserves **Sign**-pullbacks along the $\mathcal{M}^{\mathbf{Sign}}$-morphism d, there is $\varphi_{B'} \in \mathbf{Sen}(\Sigma_{B'})$ with $b(\varphi_{B'}) = \varphi_B$ and $h'(\varphi_{B'}) = \varphi_{D'}$.

$$\begin{tikzcd}
\{*\} \arrow[dr, dotted, "\varphi_{B'}"] \arrow[drr, bend left, "\varphi_{D'}"] \arrow[ddr, bend right, "\varphi_B"'] & & \\
& \mathbf{Sen}(\Sigma_{B'}) \arrow[r, "h'"] \arrow[d, tail, "b"] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbf{Sen}(\Sigma_{D'}) \arrow[d, tail, "d"] \\
& \mathbf{Sen}(\Sigma_B) \arrow[r, "h"] & \mathbf{Sen}(\Sigma_D)
\end{tikzcd}$$

Since $b \in \mathcal{M}^{\mathbf{Pres}}$, $\varphi_{B'} \in \Gamma_{B'}^\bullet$. Hence $h'(\varphi_{B'}) = \varphi_{D'} \in \Gamma_{D'}^\bullet$. Finally, the front faces are pullbacks in **Pres** because they are so in **Sign** and $b, c, d \in \mathcal{M}^{\mathbf{Pres}}$.

For the "if" direction, assume that the front faces are pullbacks in **Pres** and $d \in \mathcal{M}^{\mathbf{Pres}}$. We need to show that the top face is a pushout in **Pres**. Since pushouts in **Sign** are hereditary, it is a pushout in **Sign**. It remains to show that $\Gamma_{D'} \models\!\mid h'(\Gamma_{B'}) \cup k'(\Gamma_{C'})$. The "$\models$" direction follows from h' and k' being presentation morphisms. Concerning the "$=\!\mid$" direction, let $\varphi_{D'} \in \Gamma_{D'}$. Since d is a presentation morphism, $d(\varphi_{D'}) \in \Gamma_D^\bullet$. Since the bottom face is a pushout in **Pres**, $d(\varphi_{D'}) \in (h(\Gamma_B) \cup k(\Gamma_C))^\bullet$. Since $b, c \in \mathcal{M}^{\mathbf{Pres}}$, the latter term is $(h(b(\Gamma_{B'})) \cup k(c(\Gamma_{C'})))^\bullet = (d(h'(\Gamma_{B'})) \cup d(k'(\Gamma_{C'})))^\bullet = d(h'(\Gamma_{B'}) \cup k'(\Gamma_{C'}))^\bullet$. Since $d \in \mathcal{M}^{\mathbf{Pres}}$, we obtain $\varphi_{D'} \in (h'(\Gamma_{B'}) \cup k'(\Gamma_{C'}))^\bullet$.

Finally, we need to show that upper adjoints of inverse images exist in **Pres**, assuming the same for **Sign**.

$$\begin{tikzcd}
& A \arrow[r, "f"] & B & \\
C \arrow[ur, tail, "m"] & C' \arrow[l, tail, "\iota"] \arrow[u, tail, "f^{-1}(n)"] \arrow[r, "g"'] \arrow[ur, phantom, "\urcorner", very near start] & D \arrow[u, tail, "n"] \arrow[r, tail, "\kappa"'] & D' \arrow[ul, tail, "\forall_f(m)"']
\end{tikzcd}$$

We first need to show that $\forall_f$ acts on $\mathcal{M}^{\mathbf{Pres}}$. Given $m \in \mathcal{M}^{\mathbf{Pres}}$, we define

$$\Gamma_{D'} = \forall_f(m)^{-1}(\Gamma_B^{\bullet}),$$

which turns $\forall_f(m)$ into a conservative presentation morphism.

Second, in order to show that $f^{-1}(n) \sqsubseteq m$ iff $n \sqsubseteq \forall_f(m)$ in **Pres**, we can assume it for **Sign**. All we then need to show is that in case that both hand-sides hold in **Sign** (implying that ι and κ exist as signature morphisms), then ι is a conservative presentation morphism iff κ is. But ι and κ are always both presentation morphisms because m and $\forall_f(m)$ are conservative, and ι and κ are always both conservative because $f^{-1}(n)$ and n are. $\square$

4. Nonconservative presentation morphisms

Our central result Theorem 4 assumes that we work with conservative presentation morphisms. However, in some applications of generic filtering, it is useful to also filter out axioms, see section 1.3. So the question arises whether Theorem 4 can be generalised to the nonconservative case: does the theorem still hold if the admissible class of monos $\mathcal{M}^{\mathbf{Pres}}$ is taken to be the class of *all* (i.e. not necessarily conservative) presentation morphisms with underlying signature morphism in $\mathcal{M}^{\mathbf{Sign}}$?

Unfortunately, the answer is no. Consider the institution with one signature Σ (and the identity signature morphism), two Σ-sentences φ and ψ, and two Σ-models M and N such that $M \models \varphi$, $M \not\models \psi$, $N \not\models \varphi$ and $N \models \psi$. Clearly, this is a MipMap institution. Nevertheless, the category **Pres** of presentation morphisms is not MipMap when $\mathcal{M}^{\mathbf{Pres}}$ is defined as above. Namely, consider the following pushout in **Pres**:

$$\begin{tikzcd}
\{\varphi\} \arrow[r] \arrow[d] & \{\varphi, \psi\} \arrow[d] \\
\{\varphi, \psi\} \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & \{\varphi, \psi\}
\end{tikzcd}$$

Assume that this pushout were hereditary. This would mean that the upper square is a pushout in the category of partial maps:

$$\begin{tikzcd}
\{\varphi\} & \{\varphi\} \arrow[l, tail] \arrow[r] & \{\varphi,\psi\} & & \\
\{\varphi\} \arrow[u, tail] \arrow[d] & & \{\varphi,\psi\} \arrow[u, tail] \arrow[d] \arrow[drr, dotted] \arrow[ddr, dotted] & & \\
\{\varphi,\psi\} & \{\varphi,\psi\} \arrow[l, tail] \arrow[r] \arrow[drr, dotted] \arrow[ddr, dotted] & \{\varphi,\psi\} \arrow[ul, phantom, "\ulcorner", very near start] & & \{\varphi\} \arrow[uull, tail] \arrow[dd] \\
 & & & \bullet \arrow[ul, dashed] \arrow[dr, dashed] & \\
 & & \{\varphi\} \arrow[uull, tail] \arrow[rr] & & \{\varphi,\psi\}
\end{tikzcd}$$

Then for the cocone of partial maps shown in the lower right corner of the diagram, a mediating partial map (shown as dashed morphisms) would exist. Since composition of partial maps is given by pullback, the dotted arrows would span two pullbacks. However, because pullbacks of inclusions are intersections, the pullback object would need to be $\{\varphi\}$ — a contradiction. □

5. Conclusion

We have given a semantics to generic specifications with filtering, combining the pushouts-style semantics of generics in CASL with filtering (i.e. syntactic removal of parts of a specification) in DOL. Heindel's notion of MipMap category can be extended to the notion of MipMap institution. We have show that in a MipMap institution, the category of presentations and conservative presentation morphisms is MipMap as well. This provides a semantics of generic specifications with filtering via pushouts of partial presentation morphisms.

The case of filtering along nonconservative presentation morphisms (that is, deletion of axioms) is not covered by our theory. However, it could be handled by constructing the pushout signature and presentation in the same way as in the conservative case, but without guarantee that the pushout property holds.

Tool support for CASL and DOL is provided by the *Heterogeneous Tool Set*, Hets [19]. Future work will extend this to GDOL.

References

[1] Astesiano, E., Bidoit, M., Krieg-Brückner, B., Kirchner, H., Mosses, P.D., Sannella, D., Tarlecki, A.: CASL - the Common Algebraic Specification Language. Theoretical Computer Science 286, 153–196 (2002), `http://www.cofi.info`

[2] Baumeister, H., Cerioli, M., Haxthausen, A., Mossakowski, T., Mosses, P.D., Sannella, D., Tarlecki, A.: Casl semantics. In: Mosses, P.D. (ed.) CASL Reference Manual, Lecture Notes in Computer Science, vol. 2960, part ÏII. Springer Verlag, London (2004), http://www.springerlink.com/(bt4qw245oavupgzdxw3zpuul)/app/home/contribution.asp?referrer=parent&backto=searchcitationsresults,25,38;, ÃŃdited by D. Sannella and A. Tarlecki

[3] Bidoit, M., Mosses, P.D. (eds.): CASL User Manual, Lecture Notes in Computer Science, vol. 2900. Springer, Berlin, Heidelberg (2004)

[4] Cazanescu, V.E., Rosu, G.: Weak inclusion systems. Mathematical Structures in Computer Science 7(2), 195–206 (1997), https://doi.org/10.1017/S0960129596002253

[5] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2006), https://doi.org/10.1007/3-540-31188-2

[6] Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic approaches to graph transformation - part II: single pushout approach and comparison with double pushout approach. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations, pp. 247–312. World Scientific (1997)

[7] Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specification 1: Equations und Initial Semantics, EATCS Monographs on Theoretical Computer Science, vol. 6. Springer (1985), https://doi.org/10.1007/978-3-642-69962-7

[8] Goguen, J.A., Burstall, R.M.: Institutions: Abstract model theory for specification and programming. Journal of the Association for Computing Machinery 39, 95–146 (1992)

[9] Hayman, J., Heindel, T.: On pushouts of partial maps. In: Giese, H., König, B. (eds.) Graph Transformation - 7th International Conference, ICGT 2014, Held as Part of STAF 2014, York, UK, July 22-24, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8571, pp. 177–191. Springer (2014)

[10] Heindel, T.: Adhesivity with partial maps instead of spans. Fundam. Inform. 118(1-2), 1–33 (2012), https://doi.org/10.3233/FI-2012-704

[11] Horrocks, I., Kutz, O., Sattler, U.: The Even More Irresistible $\mathcal{SROIQ}$. In: Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2006). pp. 57–67. AAAI Press (June 2006)

[12] Kennaway, R.: Graph rewriting in some categories of partial morphisms. In: Ehrig, H., Kreowski, H., Rozenberg, G. (eds.) Graph-Grammars and Their Application to Computer Science, 4th International Workshop, Bremen, Germany, March 5-9, 1990, Proceedings. Lecture Notes in Computer Science, vol. 532, pp. 490–504. Springer (1990), `https://doi.org/10.1007/BFb0017408`

[13] Krieg-Brückner, B.: Generic Ontology Design Patterns: Qualitatively Graded Configuration. In: Lehner, F., Fteimi, N. (eds.) KSEM 2016, The 9th International Conference on Knowledge Science, Engineering and Management. Lecture Notes in Artificial Intelligence, vol. 9983, pp. 580–595. Springer International Publishing (2016)

[14] Krieg-Brückner, B., Mossakowski, T.: Safe Ontology Development with Generic Ontology Design Patterns. In: 8th Workshop on Ontology Design and Patterns - WOP2017 (to appear)

[15] Krieg-Brückner, B., Sannella, D.: Structuring specifications in-the-large and in-the-small: Higher-order functions, dependent types and inheritance in SPECTRAL. In: Abramsky, S., Maibaum, T.S.E. (eds.) TAPSOFT '91: Proceedings of the International Joint Conference on Theory and Practice of Software Development Brighton, UK, April 8–12, 1991. pp. 313–336. Springer Berlin Heidelberg, Berlin, Heidelberg (1991), `http://dx.doi.org/10.1007/3540539816\_74`

[16] Lucanu, D., Li, Y.F., Dong, J.S.: Semantic Web Languages—Towards an Institutional Perspective. In: Futatsugi, K., Jouannaud, J.P., Meseguer, J. (eds.) Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday. Lecture Notes in Computer Science, vol. 4060, pp. 99–123. Springer (2006)

[17] Mossakowski, T., Codescu, M., Neuhaus, F., Kutz, O.: The Distributed Ontology, Modeling and Specification Language – DOL. In: Koslow, A., Buchsbaum, A. (eds.) The Road to Universal Logic, vol. I, pp. 489–520. Birkhäuser (2015)

[18] Mossakowski, T., Kutz, O., Codescu, M., Lange, C.: The distributed ontology, modeling and specification language. In: Vescovo, C.D., Hahmann, T., Pearce, D., Walther, D. (eds.) WoMo 2013. CEUR-WS online proceedings, vol. 1081 (2013)

[19] Mossakowski, T., Maeder, C., LÃijttich, K.: The heterogeneous tool set, hets. In: Grumberg, O., Huth, M. (eds.) TACAS. Lecture Notes in Computer Science, vol. 4424, pp. 519–522. Springer (2007), `http://dblp.uni-trier.de/db/conf/tacas/tacas2007.html#MossakowskiML07`

[20] Mosses, P.D. (ed.): CASL Reference Manual, Lecture Notes in Computer Science, vol. 2960. Springer, Berlin, Heidelberg (2004)

[21] Object Management Group: The distributed ontology, modeling, and specification language (DOL) (2016), OMG standard available at `omg.org/spec/DOL`. See also `dol-omg.org`

[22] Robinson, E., Rosolini, G.: Categories of partial maps. Inf. Comput. 79(2), 95–130 (1988), `https://doi.org/10.1016/0890-5401(88)90034-X`