Partial Pushout Semantics of Generics in GDOL

Till Mossakowski[1] Bernd Krieg-Brückner[2]

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

[1]University of Magdeburg

[2]University of Bremen

IFIP WG 1.3 meeting, 2017-09-05

Overview

Overview

Specification & Graph Transformation

$$
\begin{tikzcd}
\text{Category theory} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner" very near end] & \begin{array}{c}\text{Pushout-semantics of}\\ \text{graph transformation}\end{array} \arrow[d, dotted] \\
\begin{array}{l}\text{Institution semantics of}\\ \text{structured specification}\end{array} \arrow[r, dotted] & \color{red}{\text{GDOL with generics and filtering}}
\end{tikzcd}
$$

CASL and DOL

- Common Algebraic Specification Language (CASL)
  - initiated, designed and approved by (members of) IFIP WG 1.3
  - subsorted partial first-order logic with induction
  - powerful constructs for structured and architectural specification
    - institution independent semantics
- Distributed Ontology, Model, and Specification Language (DOL)
  - standardised by Object Management Group (OMG)
  - extends CASL
    - filtering, uniform interpolation, circumscription, ...
  - conformant logics:
    OWL, FOL/TPTP, CASL, Common Logic, UML

Generics & Filtering

$$
\begin{tikzcd}
\text{CASL without generics} \arrow[r] \arrow[d] & \text{CASL with generics} \arrow[d, dotted] \\
\text{DOL with filtering} \arrow[r, dotted] & \textcolor{red}{\text{GDOL with generics and filtering}} \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}
$$

Generic specification of lists 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
```

Instantiation of lists in CASL

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

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

DOL Filtering

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 Nat
end
```

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

Result of DOL Filtering

```
spec List_filtered[sort Elem] =
  free type List[Elem] ::=
      [] | __::__(Elem; List[Elem])
end
```

Generic specification of finite sets 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
```

DOL Filtering, Again

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

Generic Filtering for Containets

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.

Generic Ontology Design Patterns

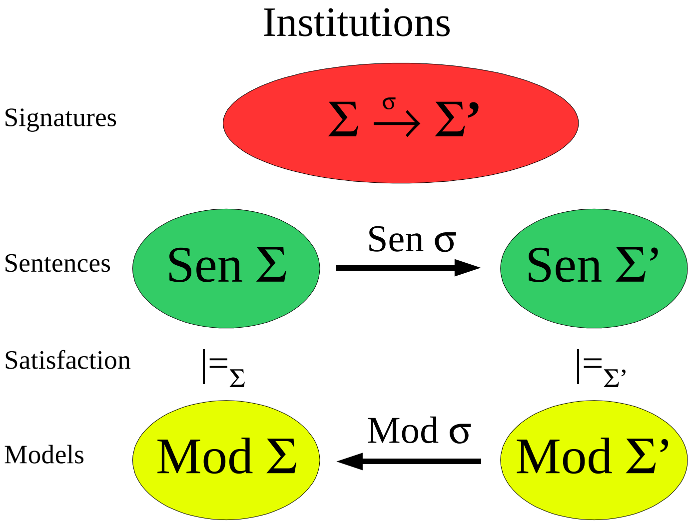
Consider a pattern for subclasses Z1 and Z2 of X with a disjoint union axiom for the subclasses
We use the Web Ontology Language OWL with Manchester syntax.

```
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.

Overview

Institutions (intuition)



Institutions (formal definition)

An institution $\mathcal{I} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \langle \models_\Sigma \rangle_{\Sigma \in |\mathbf{Sign}|} \rangle$ consists of:

- a category **Sign** of signatures;
- a functor $\mathbf{Sen} \colon \mathbf{Sign} \to \mathbf{Set}$,giving a set $\mathbf{Sen}(\Sigma)$ of Σ-sentences for each signature $\Sigma \in |\mathbf{Sign}|$, and a function $\mathbf{Sen}(\sigma) \colon \mathbf{Sen}(\Sigma) \to \mathbf{Sen}(\Sigma')$ that yields σ-translation of Σ-sentences to Σ'-sentences for each $\sigma \colon \Sigma \to \Sigma'$;
- a functor $\mathbf{Mod} \colon \mathbf{Sign}^{op} \to \mathbf{Set}$,giving a set $\mathbf{Mod}(\Sigma)$ of Σ-models for each signature $\Sigma \in |\mathbf{Sign}|$, and a functor $red\_\sigma = \mathbf{Mod}(\sigma) \colon \mathbf{Mod}(\Sigma') \to \mathbf{Mod}(\Sigma)$; for each $\sigma \colon \Sigma \to \Sigma'$;
- for each $\Sigma \in |\mathbf{Sign}|$, a satisfaction relation $\models_{\mathcal{I},\Sigma} \subseteq \mathbf{Mod}(\Sigma) \times \mathbf{Sen}(\Sigma)$

such that for any signature morphism $\sigma \colon \Sigma \to \Sigma'$, Σ-sentence $\varphi \in \mathbf{Sen}(\Sigma)$ and Σ'-model $M' \in \mathbf{Mod}(\Sigma')$:

$$M' \models_{\mathcal{I},\Sigma'} \sigma(\varphi) \iff redM'\sigma \models_{\mathcal{I},\Sigma} \varphi \qquad \text{[Satisfaction condition]}$$

Institutions: examples

- Many-sorted first-order logic with equality $\mathbf{MSFOL}^=$.
- Many-sorted first-order logic with equality and sort generation constraints $\mathbf{MSCFOL}^=$
- Description Logics / Web Ontology Language OWL

Working in an arbitrary but fixed institution

logical consequence:

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

iff for all Σ-models M, we have

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

theory closure: $\Gamma^\bullet := \{\varphi \in \mathbf{Sen}(\Sigma) \mid \Gamma \models \varphi\}$

$\Gamma_1, \Gamma_2 \subseteq \mathbf{Sen}(\Sigma)$ are logically equivalent, written $\Gamma_1 \models\!\mid \Gamma_2$, if $\Gamma_1 \models \Gamma_2$ and $\Gamma_2 \models \Gamma_1$ (or equivalently, $\Gamma_1^\bullet = \Gamma_2^\bullet$).

Presentations

presentation: 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 $\sigma\colon \langle \Sigma, \Gamma \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ are those signature morphisms $\sigma\colon \Sigma \longrightarrow \Sigma'$ for which $\Gamma' \models_{\Sigma'} \sigma(\Gamma)$, or, in other words $\sigma(\Gamma) \subseteq \Gamma'^{\bullet}$.

This gives a category **Pres** of presentations

$\sigma\colon \langle \Sigma, \Gamma \rangle \longrightarrow \langle \Sigma', \Gamma' \rangle$ is conservative, if $\Gamma \models \sigma^{-1}(\Gamma')$.

Overview

Pushout Semantics of Generic Specifications

Pushout-style semantics [EhrigMahr85, CASLBooks]:

- generic specification = $PAR \to BODY$ in **Pres**
- application to argument specification ARG with fitting morphism $PAR \to ARG$ as 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.

Double-Pushout Approach

The possibility of deleting symbols from the parameter requires a more general situation like this:

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

Resembles the double-pushout approach form algebraic graph transformation.

(Weak) Adhesive HLR Categories

In a weak adhesive HLR category, we have:

Theorem (Uniqueness of pushout complements [EhrigBook06])

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.

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

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

$$PAR \longleftarrow INTERFACE \longrightarrow BODY$$

to an argument ARG is then given by

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

Problem with Glueing Condition

```
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]
```

leads to the following diagram:

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

Cannot be complemented to a pushout, because the operation **op** `g:s->t` is "dangling": the sort `t` is missing in the interface.

Overview

Single-Pushout Approach

Single-pushout (SPO) approach to graph transformation with pushouts in a category of partial maps:

$$\begin{tikzcd}
PAR \arrow[r, "\circ" description] \arrow[d] & BODY \arrow[d] \\
ARG \arrow[r, "\circ" description] & RESULT
\end{tikzcd}$$

Infrastructure for Partial Maps

Definition (Admissible class of monos [RobinsonRosolini88])

A class $\mathcal{M}$ of monos in a category $\mathbf{C}$ is admissible, if

$\mathbf{C}$ has pullbacks along $\mathcal{M}$-morphisms,

$\mathcal{M}$ is stable under pullback,

$\mathcal{M}$ contains all identities, and

$\mathcal{M}$ is closed under composition.

Example: if $\mathbf{C}$ has pullbacks, then $\mathcal{M}$ = all monos is admissible.

For $A \in |\mathbf{C}|$, define lattice $(Sub_{\mathcal{M}}A, \sqsubseteq)$ of subobjects of A:

- equivalence classes $[m]$ of morphisms $m : M \hookrightarrow A \in \mathcal{M}$
- taken up to isomorphism in the comma category $(\mathbf{C}, A)$
- ordering $m \sqsubseteq n$ if there exists some i with $m = i; n$.

Category of Partial Maps

Definition (Category of partial maps [RobinsonRosolini88])

The category $\mathbf{C}_{*\mathcal{M}}$ of $\mathcal{M}$-partial maps has objects as in $\mathbf{C}$

- Morphisms from $[(m, dom, f)] : A \to B$ are spans $A \xleftarrow{m} dom \xrightarrow{f} B$ in $\mathbf{C}$ with $m \in \mathcal{M}$, taken up to isomorphism of spans
- composition = pulling back

$$\begin{tikzcd} & dom \arrow[r] \arrow[d, tail] \arrow[dr, phantom, "\lrcorner", very near start] & dom_2 \arrow[r, "f_2"] \arrow[d, tail, "m_2"] & C \\ A & dom_1 \arrow[l, tail, "m_1"'] \arrow[r, "f_1"] & B & \end{tikzcd}$$

$[(m, X, f)]$ is total if m is an isomorphism.

Embedding functor $\Gamma : \mathbf{C} \to \mathbf{C}_{*\mathcal{M}}$ $\quad f : A \to B \mapsto [(id_A, A, f)] : A \to B$

Inverse Images

Definition ([RobinsonRosolini88])

$f^{-1} : Sub_{\mathcal{M}}B \to Sub_{\mathcal{M}}A$

takes $[m] \in Sub_{\mathcal{M}}B$ to $[m'] \in Sub_{\mathcal{M}}A$ given by a pullback

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

Upper Adjoint to Inverse Image

Definition ([HaymanHeindel14])

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 $\forall_f$.

Existence of Pushouts of Partial Maps

Definition ([Kennaway90])

A pushout in $\mathbf{C}$ is hereditary if Γ maps it to a pushout in $\mathbf{C}_{*\mathcal{M}}$.

Theorem ([HaymanHeindel14])

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.

Overview

MipMap Institutions

Definition (MipMap category [HaymanHeindel14])

A category **C** with an admissible class of monos $\mathcal{M}$ is called a MipMap category, if **C** has hereditary pushouts and upper adjoints of inverse images.

Theorem ([HaymanHeindel14])

Each topos is MipMap.

Definition (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 $\mathbb{S}et$.

Sample MipMap institutions

- OWL with monos: signature category is $\mathbb{S}et^3$
- **MSFOL**$^=$ with monos: signature category is a topos
- **MSCFOL**$^=$: dto.

Central Theorem

Theorem

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.

Characterisation of hereditary pushouts [Heindel12]

Theorem (POs: hereditary iff partial van-Kampen square)

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

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

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

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

Nonconservative Presentation Morphisms

- needed to filter out axioms
- can main theorem be generalised to the nonconservative case?

A Counterexample

Institution with

- one signature Σ
- two Σ-sentences φ and ψ
- two Σ-models M and N
- $M \models \varphi$, $M \not\models \psi$, $N \not\models \varphi$ and $N \models \psi$

The is a MipMap institution.

But the category **Pres** of presentation morphisms with $\mathcal{M}^{\mathbf{Pres}}$= all monos is not MipMap!

Consider the following pushout in **Pres**:

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

A Counterexample (cont'd)

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

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

Conclusion

Conclusion

- Notion of MipMap institution
- category of presentations and conservative presentation morphisms is MipMap as well
- Provides a semantics of generic specifications with filtering via pushouts of partial presentation morphisms.

Future work

- Filtering along nonconservative presentation morphisms: use same construction, but without guarantee that the pushout property holds?
- Tool support for CASL and DOL is provided by Hets: extend this to GDOL.