2-Term Charters

Alexander Knapp and María Victoria Cengarle
Universität Augsburg, Germany

ABSTRACT Building on Martin Gogolla's algebraic formalisation of the "Object Constraint Language" (OCL), its expression part has been formalised as a syntactically and semantically heterogeneous language using the framework of term charters for faithfully integrating OCL's features in a step-wise and compositional manner. This schema for the evaluation of OCL expressions on a single system state is now extended to comprise OCL contracts, i.e., the specification of operations with pre- and post-conditions. Under mild assumptions term charters can be systematically transformed into 2-term charters involving two system states. The application to OCL as a heterogeneous expression and contract language is illustrated by several examples.

KEYWORDS Object Constraint Language, Term charters, Institutions.

1. Introduction

Martin Gogolla has been involved in the foundations, the development, and the improvement of the "Object Constraint Language" (OCL) since its very inception and its integration with the "Unified Modeling Language" (UML) (Gogolla & Richters 1997; Richters & Gogolla 1998). He has contributed and continually updated a formal semantics for OCL that still is at the heart of the (informative) semantics annex of the official OMG specification (Object Management Group 2014). He has furthermore developed the Bremen "UML-based Specification Environment" (USE) that, based on the OCL and its semantics, allows for UML model exploration, animation, visualisation, testing, finding, completion, to name a few (Gogolla & Richters 2002; Gogolla et al. 2007, 2020). USE continues to be a benchmark implementation of the OCL specification.

Martin's work on semantics and the interaction with him at several OCL workshops and UML conferences were a main source of inspiration for our own efforts on providing OCL with an operational as well as a denotational semantics (Cengarle & Knapp 2001, 2004). Martin's OCL semantics and the USE tool follow the tradition of algebraic specifications that, in fact, stood at Martin's career beginnings (Engels & Gogolla 1982; Ehrich et al. 1989). Also in this algebraic line of research, we have started to restructure our semantics using an algebraic term evaluation framework akin to the model-theoretic framework of institutions (Knapp & Cengarle 2015). The goal of this endeavour is to show how the various semantic and syntactic features of OCL can be handled in a modular and compositional manner.

In our framework of term charter domains and term charters (Knapp & Cengarle 2018), the domain of evaluation describes values and variables, structures, and the underlying values of structures, all indexed over signatures. A term charter consists of a term constructor, a variable embedding, and evaluation maps for terms over a signature and variables; the interaction of these ingredients is regulated by three axioms for variable evaluation, variable renaming, and signature translations. Analogous to institutions (Goguen & Burstall 1992), the slogan behind term charters is "evaluation is invariant under change of notation". Moreover, term charters give rise to institutions and thus a formulation of OCL by means of term charters makes it possible to integrate also this language into an institution-based modularisation of the UML as a heterogeneous language (Cengarle et al. 2008). Furthermore, for modular language design, term charter domains and term charters can be combined systematically keeping up the invariance of evaluation under change of notation (Knapp & Cengarle 2015, 2018).

The OCL offers, besides an expression language for querying and navigating through a system state (snapshot), a contract language for specifying the behaviour of operations by pre- and

JOT reference format:
Alexander Knapp and María Victoria Cengarle. *2-Term Charters*. Journal of Object Technology. Vol. 19, No. 3, 2020. Licensed under Attribution - NonCommercial - No Derivatives 4.0 International (CC BY-NC-ND 4.0)
http://dx.doi.org/10.5381/jot.2020.19.3.a10

post-conditions. The pre-condition of a contract is a boolean OCL expression that is to be evaluated over the operation's pre-state; the evaluation of the post-condition expression, however, involves two system states, the pre-state and a post-state where the suffix `@pre` refers to the pre-state. We demonstrate how expression evaluation over two states can be accommodated for by the notion of term charter domains and term charters by devising a suitable notion of "2-able" term charter domains, 2-term charter domains, and 2-term charters.

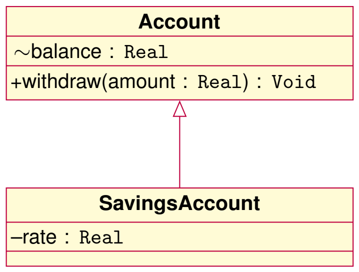
Technically, we use the language of indexed categories (Tarlecki et al. 1991) for our general abstract framework and also for the construction of 2-term charter domains and of 2-term charters. We show that, to ensure that a given term charter can be transformed into a 2-term charter, it suffices that the signatures, used as indexes, have push-outs and that the structures support a weak form of amalgamation (Sannella & Tarlecki 2012). In fact, a 2-term charter also is a standard term charter yet over a 2-term charter domain. Thus, the term-charter-building operators can also be applied to 2-term charters.

The extension from single states to pairs of states, in order to cover also state transitions, has been considered before in different algebraic contexts though mainly on the level of formulæ and satisfaction and not for expressions and evaluation. In particular, Baumeister (1995) studies the relational language of Z in terms of institutions. Similarly, Pawłowski (1996) presents context institutions for capturing open formulæ and their usage in Hoare logic. For temporal logic and whole sequences of states, Reggio, Astesiano, and Choppy extend the algebraic specification language CASL by a linear temporal logic (Reggio et al. 2003).

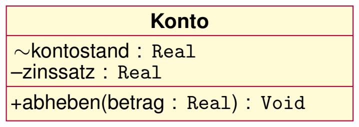
The remainder of this paper is structured as follows: We first give a rough account of a small sub-language of OCL expressions in Sect. 2 that shows different language features and their semantic formalisation. In Sect. 3, we provide an abstract account of the ingredients in the framework of term charter domains and term charters, and illustrate the achieved modularisation by several examples. In Sect. 4, we develop our approach of extending OCL expression evaluation over a single system state to an evaluation over two system states for capturing OCL contracts and derive requirements on "2-able" term charter domains. In Sect. 5, we then give an abstract account of 2-term charter domains and 2-term charters and show how these can again be extended in a modular fashion. We conclude in Sect. 6 with an outlook to future work.

2. OCL Types and Expressions

We give a brief account of what is involved when formalising the type system of OCL and its expressions, which can be used to navigate and constrain single system states. The main goal of this rather abridged overview is not completeness or an in-depth discussion of the fine points of, say, non-determinism or three-valued evaluation (Cengarle & Knapp 2004; Brucker & Wolff 2008; Object Management Group 2014). The aim is more to give an impression of how the semantics can be captured using extended order-sorted algebras, to show some points of variation, and also to show why it is useful to proceed to a more general framework than that immediately provided by algebras. We also stress the importance of maps between different systems, system states, and OCL expressions for making them usable in systematic software development.



(a) English



(b) German

Figure 1 UML class diagrams for accounts

2.1. OCL Types and States

The OCL types and operations comprise, on the one hand, predefined types, like `Real` or `Seq`(s), together with their functions, like `+`: `Real` $\times$ `Real` $\to$ `Real` or `append`: `Seq`$(s) \times$ `Seq`$(s) \to$ `Seq`(s), of the standard library and, on the other hand, the classes, attributes (properties), and queries of an underlying UML static structure. These types and operations can be captured — at least to a large extent — by an *order-sorted signature* $\Sigma = (S, F)$ consisting of a partial order $S = (|S|, \leq_S)$ for the sort (class, type) hierarchy and a family of function declarations $F = (F_{\bar{s}})_{\bar{s} \in |S|^+}$. For complying with the OCL specification such signatures have to be *closed* under the OCL primitives and they must reflect the OCL type hierarchy with its collection types, such that, e.g., $\mathtt{Real} \in |S|$, $\mathtt{Seq}(s), \mathtt{Collection}(s) \in |S|$, and $\mathtt{Seq}(s) \leq_S \mathtt{Collection}(s)$ for all $s \in |S|$. For example, the description of accounts in Fig. 1(a) leads to the OCL-closed sort hierarchy $(\mathrm{S}^{\mathrm{Accs}}, \leq_{\mathrm{Accs}})$ containing

$$\begin{aligned}
\mathrm{S}^{\mathrm{Accs}} &= \{\, \textsf{Account}, \mathtt{Seq}(\textsf{Account}), \textsf{SavingsAccount}, \\
&\qquad \mathtt{Void}, \mathtt{Real}, \ldots \,\}\,, \\
&\textsf{SavingsAccount} \leq_{\mathrm{Accs}} \textsf{Account}\,, \\
&\mathtt{Seq}(\textsf{SavingsAccount}) \leq_{\mathrm{Accs}} \mathtt{Seq}(\textsf{Account})
\end{aligned}$$

and the function declarations $\mathrm{F}^{\mathrm{Accs}}$ including

$$\begin{aligned}
&\mathrm{F}^{\mathrm{Accs}}_{\mathtt{Real}} = \{\, \mathtt{-1.0}, \mathtt{0.0}, \mathtt{3.14159265359}, \ldots \,\}\,, \\
&\mathrm{F}^{\mathrm{Accs}}_{\mathtt{Real}\,\mathtt{Real}\,\mathtt{Real}} = \{\, \mathtt{+}, \ldots \,\}\,, \\
&\mathrm{F}^{\mathrm{Accs}}_{\textsf{Account}\,\mathtt{Real}} = \{\, \textsf{balance} \,\}\,, \\
&\mathrm{F}^{\mathrm{Accs}}_{\textsf{SavingsAccount}\,\mathtt{Real}} = \{\, \textsf{balance}, \textsf{rate} \,\}\,,
\end{aligned}$$

where the declaration of balance for a SavingsAccount is due to the rules of inheritance in UML. The operations, i.e., withdraw in our example, are not captured in this kind of signature.

Changes in the interface of UML static structures, i.e., moving from an order-sorted signature $\Sigma = (S, F)$ to an order-sorted signature $\Sigma' = (S', F')$, are catered for by *order-sorted signature morphisms* $\sigma = (\sigma_S \colon S \to S', \sigma_F \colon F \to F') \colon \Sigma \to \Sigma'$ where σ_S is monotone w.r.t. $\leq_S$ and $\sigma_F(f) \in F'_{\sigma_S(\bar{s})}$ for $f \in F_{\bar{s}}$. For example,

$$\sigma_S^{\mathsf{Accs}}(\mathsf{Account}) = \sigma_S(\mathsf{SavingsAccount}) = \mathsf{Konto}\ ,$$
$$\sigma_F^{\mathsf{Accs}}(\mathsf{balance}) = \mathsf{kontostand}\ ,$$
$$\sigma_F^{\mathsf{Accs}}(\mathsf{rate}) = \mathsf{zinssatz}$$

allows to use a German interface to accounts, at the same time discarding the difference between checking and savings accounts. It is important to require that all the OCL standard types and operations are left invariant by such signature morphisms, that is, e.g., $\sigma_S(\mathtt{Real}) = \mathtt{Real}$ and $\sigma_S(\mathtt{Seq}(s)) = \mathtt{Seq}(\sigma_S(s))$.

A *system state* or *snapshot* for a given static structure expressed as an order-sorted signature $\Sigma = (S, F)$ can be represented by an order-sorted Σ-*algebra* $M = (S^M, F^M)$ interpreting the sort hierarchy by a family of sets $S^M = (s^M)_{s \in |S|}$ such that $s_1^M \subseteq s_2^M$ if $s_1 \leq_S s_2$, and the function declarations by a family of (total) functions $F^M = (F_{\bar{s}}^M)_{\bar{s} \in |S|^+}$ with $f^M \colon s_1^M \times \ldots \times s_n^M \to s^M \in F^M_{s_1 \ldots s_n s}$ for $f \in F_{s_1 \ldots s_n s}$. The interpretation of a sort from a UML class corresponds to the objects of this class in the system state of discourse. Such a set may be an arbitrary, typically finite set or it may be a subset of an externally given set *Obj* of object identifiers. For a predefined OCL type, a standard interpretation has to be required, such that, e.g., the interpretation $\mathtt{Bool}^M$ of the sort $\mathtt{Bool}$ always yields the Boolean values $\{\mathit{ff}, \mathit{tt}\}$ and the interpretation $\mathtt{Seq}(s)^M$ of the sequences over s always yields the lists $(s^M)^*$. Moreover, to comply with OCL 2.4, both a "null" value $\mathtt{0}$ and an "invalid" value $\dagger$ have to be included in each sort interpretation (where $\mathtt{0}$ can occur in collections, but $\dagger$ cannot); in older OCL versions, $\dagger$ alone would have sufficed. An example state St for Accs may thus contain

$$\mathtt{Real}^{\mathsf{St}} = \{\, \dagger,\ \mathtt{0},\ -1.0,\ 0.0,\ 3.14159265359,\ \ldots \,\}\ ,$$
$$\mathsf{Account}^{\mathsf{St}} = \{\, \dagger,\ \mathtt{0},\ 3048,\ 4711 \,\}\ ,$$
$$\mathsf{SavingsAccount}^{\mathsf{St}} = \{\, \dagger,\ \mathtt{0},\ 3048 \,\}\ .$$

The functions interpreting the attributes and query operations of the UML static structure again may be chosen at whim, whereas the interpretation of the operations from the OCL standard library should indeed be standard, such that $+^M(0.0, 0.0) = 0.0$, $+^M(0.0, \mathtt{0}) = \dagger$, and $\mathtt{append}^M(0.0 :: 1.0, \varepsilon) = 0.0 :: 1.0$ with $\mathtt{Seq\{\}}^M = \varepsilon$. The state St may hence include

$$\mathsf{balance}^{\mathsf{St}}(\dagger) = \dagger = \mathsf{balance}^{\mathsf{St}}(\mathtt{0})\ ,$$
$$\mathsf{balance}^{\mathsf{St}}(3048) = 200.0\ , \quad \mathsf{balance}^{\mathsf{St}}(4711) = 100.0\ .$$

Maps between system states M_1 and M_2 over an order-sorted signature $\Sigma = (S, F)$ are given by Σ-*algebra homomorphisms* $\mu = (\mu_s \colon s^{M_1} \to s^{M_2})_{s \in |S|} \colon M_1 \to M_2$ with $\mu_s(f^{M_1}(\vec{v})) = f^{M_2}(\mu_{\bar{s}}(\vec{v}))$ for $f \in F_{\bar{s}\,s}$, where OCL standard types and operations have to be related by identities. On the other hand, an order-sorted signature morphism $\sigma \colon \Sigma \to \Sigma'$ induces a *reduct functor* $-|\sigma$ mapping a Σ'-algebra M' to the Σ-algebra $M'|\sigma = ((\sigma_S(s))^{M'})_{s \in |S|}, ((\sigma_F(F)_{\sigma_S(\bar{s})})^{M'})_{\bar{s} \in |S|^+})$ and a Σ'-algebra homomorphism $\mu' \colon M_1' \to M_2'$ to the Σ-algebra homomorphism $\mu'|\sigma = (\mu'_{\sigma_S(s)})_{s \in |S|} \colon M_1'|\sigma \to M_2'|\sigma$. For example, a system state for the static structure in Fig. 1(b) with

$$\mathsf{Konto}^{\mathsf{St}'} = \{\, \dagger,\ \mathtt{0},\ 3048 \,\}\ ,$$
$$\mathsf{kontostand}^{\mathsf{St}'}(3048) = 200.0\ ,$$
$$\mathsf{zinssatz}^{\mathsf{St}'}(3048) = 0.25$$

induces the following reduct along σ^{Accs}:

$$\begin{aligned} \mathsf{Account}^{\mathsf{St}'|\sigma^{\mathsf{Accs}}} &= \{\, \dagger,\ \mathtt{0},\ 3048 \,\} \\ &= \mathsf{SavingsAccount}^{\mathsf{St}'|\sigma^{\mathsf{Accs}}}\ , \end{aligned}$$
$$\mathsf{balance}^{\mathsf{St}'|\sigma^{\mathsf{Accs}}}(3048) = 200.0\ ,$$
$$\mathsf{rate}^{\mathsf{St}'|\sigma^{\mathsf{Accs}}}(3048) = 0.25\ .$$

2.2. OCL Expressions and Evaluation

Most OCL *expressions*, as e.g. `self.balance - 163.27` or `Seq{1.0}->isEmpty()`, correspond to *order-sorted terms* $\mathcal{T}_\Sigma^{\leq}(X) = (\mathcal{T}_\Sigma^{\leq}(X)_s)_{s \in |S|}$ over a static structure with order-sorted signature $\Sigma = (S, F)$ and Σ-*variables* $X = (X_s)_{s \in |S|}$ such that $X_{s_1} \subseteq X_{s_2}$ for $s_1 \leq_S s_2$, inductively defined for all $s \in |S|$ by

- $x \in \mathcal{T}_\Sigma^{\leq}(X)_s$ for $x \in X_s$;
- $f(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma^{\leq}(X)_{s'}$ for all $s' \geq_S s$ if $f \in F_{s_1 \ldots s_n s}$ and $t_i \in \mathcal{T}_\Sigma^{\leq}(X)_{s_i}$ for all $1 \leq i \leq n$.

Using Accs-variables X with $\mathtt{self} \in \mathrm{X}_{\mathsf{Account}}$, the order-sorted term

$$\mathtt{-}(\mathsf{balance}(\mathtt{self}),\ 163.27) \in \mathcal{T}_{\mathsf{Accs}}^{\leq}(\mathrm{X})_{\mathtt{Real}}$$

is a prefix (*abstract syntax*) variant of `self.balance - 163.27`.

The *evaluation* of the order-sorted terms $\mathcal{T}_\Sigma^{\leq}(X)$ over an order-sorted algebra $M = (S^M, F^M)$ over $\Sigma = (S, F)$ relies on a valuation

$$\beta = (\beta_s \colon X_s \to s^M) \colon X \to U_\Sigma^{\leq}(M)$$

of the variables into the *underlying values* $U_\Sigma^{\leq}(M) = S^M$. Order-sorted term evaluation

$$\begin{aligned} \llbracket - \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta) = (\llbracket - \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta)_s \colon \\ \mathcal{T}_\Sigma^{\leq}(X)_s \to s^M)_{s \in S} \colon \mathcal{T}_\Sigma^{\leq}(X) \to U_\Sigma^{\leq}(M) \end{aligned}$$

is inductively defined by

- $\llbracket x \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta)_s = \beta_s(x)$ for $x \in X_s$;
- $\llbracket f(t_1, \ldots, t_n) \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta)_s = f^M(\llbracket t_1 \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta)_{s_1}, \ldots, \llbracket t_n \rrbracket_{(\Sigma,X)}^{\leq}(M, \beta)_{s_n})$ for $f \in F_{s_1 \ldots s_n s}$ and $t_i \in \mathcal{T}_\Sigma^{\leq}(X)_{s_i}$ for $1 \leq i \leq n$.

For example, for the system state St and $\beta^{\mathrm{St}}_{\mathrm{Account}}(\mathtt{self}) = 3048$ we obtain

$$[\![\mathtt{-(balance(self),\ 163.27)}]\!]^{\leq}_{\mathrm{Accs},X}(\mathrm{St}, \beta^{\mathrm{St}}) = 36.73\ .$$

For an order-sorted signature $\Sigma = (S, F)$ *variable renamings* $\xi = (\xi_s \colon X_{1,s} \to X_{2,s})_{s \in |S|} \colon X_1 \to X_2$ allow to call `self`, say, `yo`. Such a variable renaming also induces a term renaming

$$\mathscr{T}^{\leq}_{\Sigma}(\xi) = (\mathscr{T}^{\leq}_{\Sigma}(\xi)_s \colon \mathscr{T}^{\leq}_{\Sigma}(X_1)_s \to \mathscr{T}^{\leq}_{\Sigma}(X_2)_s)_{s \in |S|} \colon \mathscr{T}^{\leq}_{\Sigma}(X_1) \to \mathscr{T}^{\leq}_{\Sigma}(X_2)\ ,$$

which usually is denoted again by ξ; then for each $t \in \mathscr{T}^{\leq}_{\Sigma}(X_1)_s$ a "renaming lemma" holds that says that a syntactic renaming of t along ξ can be equally well expressed by a semantic reshuffling of the valuation β by ξ:

($\mathrm{R}_{\leq}$) $[\![\xi_s(t)]\!]^{\leq}_{(\Sigma,X_2)}(M, \beta)_s = [\![t]\!]^{\leq}_{(\Sigma,X_1)}(M, \beta \circ \xi)_s$.

Moreover, an order-sorted signature morphism $\sigma\colon \Sigma \to \Sigma'$ and the corresponding *reducts* of variables $X'|\sigma = (X'_{\sigma_S(s)})_{s \in |S|}$ and terms $\mathscr{T}^{\leq}_{\Sigma'}(X')|\sigma = (\mathscr{T}^{\leq}_{\Sigma'}(X')_{\sigma_S(s)})_{s \in |S|}$ induce a *term translation* $\mathscr{T}^{\leq}_{\sigma}(X')\colon \mathscr{T}^{\leq}_{\Sigma}(X'|\sigma) \to \mathscr{T}^{\leq}_{\Sigma'}(X')|\sigma$ inductively defined by

- $\mathscr{T}^{\leq}_{\sigma}(X')_s(x') = x'$ for $x' \in (X'|\sigma)_s$;
- $\mathscr{T}^{\leq}_{\sigma}(X')_s(f(t_1, \ldots, t_n)) =$
  $\sigma_F(f)(\mathscr{T}^{\leq}_{\sigma}(X')_{s_1}(t_1), \ldots, \mathscr{T}^{\leq}_{\sigma}(X')_{s_n}(t_n))$
  for $f \in F_{s_1 \ldots s_n s}$ and $t_i \in \mathscr{T}^{\leq}_{\Sigma}(X'|\sigma)_{s_i}$ for $1 \leq i \leq n$.

This induced term translation is again usually denoted by just σ. Then, for the σ-reduct of a Σ'-algebra M' and a valuation $\beta'|\sigma = (\beta'_{\sigma_S(s)})_{s \in |S|}$, a "translation lemma" holds for each $t \in \mathscr{T}^{\leq}_{\Sigma}(X'|\sigma)_s$, which expresses that syntactic translations along σ correspond to semantic evaluation in reducts along σ:

($\mathrm{T}_{\leq}$) $[\![\sigma_s(t)]\!]^{\leq}_{(\Sigma',X')}(M', \beta')_{\sigma_S(s)} = [\![t]\!]^{\leq}_{(\Sigma,X'|\sigma)}(M'|\sigma, \beta'|\sigma)_s$.

As the signature and the variables form the "interface" of a term in the sense of what can be seen from the outside, the renaming lemma ($\mathrm{R}_{\leq}$) and the translation lemma ($\mathrm{T}_{\leq}$) combinedly ensure that "evaluation is invariant under change of notation".

However, several OCL constructs transcend the boundaries of mere order-sorted terms. This is most palpable for the *iteration* construct

$$c\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=}\ e_0 \mid e\mathtt{)}$$

(and its descendants like `select`, `collect`, etc.) which involves higher-order features: the iteration expression e has to be evaluated repeatedly for different valuations of the bound iteration variable i and the bound accumulator variable a. Repeated evaluation is most conveniently handled by special expression formers. This leads to an extended family of expressions $\mathscr{C}^{\mathrm{it}}_{\Sigma}(X) = (\mathscr{C}^{\mathrm{it}}_{\Sigma}(X)_s)_{s \in |S|}$ over a signature $\Sigma = (S, F)$ and Σ-variables X, which, on the one hand, satisfies $\mathscr{T}^{\leq}_{\Sigma}(X)_s \subseteq \mathscr{C}^{\mathrm{it}}_{\Sigma}(X)_s$ for all $s \in |S|$ and, on the other hand, also includes the rule

- $c\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=}\ e_0 \mid e\mathtt{)} \in \mathscr{C}^{\mathrm{it}}_{\Sigma}(X)_{s'}$
  for $s, s' \in |S|$ where $c \in \mathscr{C}^{\mathrm{it}}_{\Sigma}(X)_{\mathtt{Seq}(s)}$, $e_0 \in \mathscr{C}^{\mathrm{it}}_{\Sigma}(X)_{s'}$, and $e \in \mathscr{C}^{\mathrm{it}}_{\Sigma}(X \uplus \{i : s, a : s'\})_{s'}$

The evaluation of iterations can still be directly expressed using a Σ-algebra M based on an intermediate though ad hoc higher-order function $it^M\colon (s^M)^* \times s'^M \times (s'^M \times s^M \to s'^M) \to s^M$ that works like Haskell's foldr:

- $[\![c\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=}\ e_0 \mid e\mathtt{)}]\!]^{\mathrm{it}}_{(\Sigma,X)}(M, \beta)_{s'} =$

$$it^M([\![c]\!]^{\mathrm{it}}_{(\Sigma,X)}(M, \beta)_{\mathtt{Seq}(s)}, [\![e_0]\!]^{\mathrm{it}}_{(\Sigma,X)}(M, \beta)_{s'}, \{(v_i, v_a) \mapsto [\![e]\!]^{\mathrm{it}}_{(\Sigma,X \uplus \{i:s,a:s'\})}(M, \beta\{i : s \mapsto v_i, a : s' \mapsto v_a\})_{s'}\})\ ,$$

$$it^M(\varepsilon, v_a, h) = v_a,\ it^M(v_i :: \ell, v_a, h) = it^M(\ell, h(v_i, v_a), h).$$

Since an expression formation clause has been added, term renaming and term translation have to be extended in such a way that bounded variables are taken into account; for iterations,

- $\mathscr{C}^{\mathrm{it}}_{\Sigma}(\xi)_{s'}(c\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=} e_0 \mid e\mathtt{)}) =$

$$\mathscr{C}^{\mathrm{it}}_{\Sigma}(\xi)_{\mathtt{Seq}(s)}(c)\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=} \mathscr{C}^{\mathrm{it}}_{\Sigma}(\xi)_{s'}(e_0) \mid \mathscr{C}^{\mathrm{it}}_{\Sigma}(\xi\{i : s \mapsto i : s, a : s' \mapsto a : s'\})_{s'}(e)\mathtt{)}\ ,$$

- $\mathscr{C}^{\mathrm{it}}_{\sigma}(X')_{s'}(c\mathtt{->iterate(}i : s\mathtt{;}\ a : s' \mathtt{=} e_0 \mid e\mathtt{)}) =$

$$\mathscr{C}^{\mathrm{it}}_{\sigma}(X')_{\mathtt{Seq}(s)}(c)\mathtt{->iterate(}i : \sigma_S(s)\mathtt{;}\ a : \sigma_S(s') \mathtt{=} \mathscr{C}^{\mathrm{it}}_{\sigma}(X')_{s'}(e_0) \mid \mathscr{C}^{\mathrm{it}}_{\sigma}(X' \uplus \{i : \sigma_S(s), a : \sigma_S(s')\})_{s'}(e)\mathtt{)}\ .$$

These definitions must again ensure that "evaluation is invariant under change of notation". The renaming and the translation lemmata are adjusted as follows:

(R_{it}) $[\![\xi_s(t)]\!]^{\mathrm{it}}_{(\Sigma,X_2)}(M, \beta)_s = [\![t]\!]^{\mathrm{it}}_{(\Sigma,X_1)}(M, \beta \circ \xi)_s$

(T_{it}) $[\![\sigma_s(t)]\!]^{\mathrm{it}}_{(\Sigma',X')}(M', \beta')_{\sigma_S(s)} = [\![t]\!]^{\mathrm{it}}_{(\Sigma,X'|\sigma)}(M'|\sigma, \beta'|\sigma)_s$

Another OCL feature which is not directly included in order-sorted algebras is the possibility to access *all instances* of a finitely interpreted type, which involves a kind of reflection. Again we can capture this by a special expression formation rule this time for an extended family of expressions $\mathscr{C}^{\mathrm{a}}_{\Sigma}(X)$ extending $\mathscr{T}^{\leq}_{\Sigma}(X)$ and containing

- $s\mathtt{.allInstances()} \in \mathscr{C}^{\mathrm{a}}_{\Sigma}(X)_{\mathtt{Set}(s)}$ for $s \in |S|$

with accompanying evaluation

- $[\![s\mathtt{.allInstances()}]\!]^{\mathrm{a}}_{(\Sigma,X)}(M, \beta)_{\mathtt{Set}(s)} = \begin{cases} s^M & \text{if } |s^M| < \infty\ , \\ \dagger & \text{otherwise}\ . \end{cases}$

The evaluation of OCL expressions is *strict* to the most part, i.e., term evaluation applied to the "invalid" value $\dagger$ yields $\dagger$. Though both strict and non-strict evaluation can be directly expressed in the standard OCL order-sorted algebras containing

$\dagger$, it contributes to a separation of concerns to group together those OCL features that are to be evaluated in a non-strict fashion, viz. `isInvalid` and `if` … `then` … `else` … `endif` (as well as `and`, `or`, etc.). This arrangement results in yet another extended term construction $\mathscr{C}^{\mathrm{u}}_{\Sigma}(X)$ with according evaluation rules:

- $[\![e\texttt{.isInvalid()}]\!]^{\mathrm{u}}_{(\Sigma,X)}(M,\beta)_{\texttt{Bool}} =$
$$\begin{cases} \mathit{tt} & \text{if } [\![e]\!]^{\mathrm{u}}_{(\Sigma,X)}(M,\beta)_s = \dagger\,, \\ \mathit{ff} & \text{otherwise}\,. \end{cases}$$
- $[\![\texttt{if } e \texttt{ then } e_{\mathit{tt}} \texttt{ else } e_{\mathit{ff}} \texttt{ endif}]\!]^{\mathrm{u}}_{(\Sigma,X)}(M,\beta)_s =$
$$\begin{cases} [\![e_b]\!]^{\mathrm{u}}_{(\Sigma,X)}(M,\beta)_s & \text{if } [\![e]\!]^{\mathrm{u}}_{(\Sigma,X)}(M,\beta)_{\texttt{Bool}} = b \\ & \quad \text{with } b \in \{\mathit{ff},\mathit{tt}\}\,, \\ \dagger & \text{otherwise}\,. \end{cases}$$

Obviously, the OCL contains all of $\mathscr{C}^{\mathrm{it}}$, $\mathscr{C}^{\mathrm{a}}$, $\mathscr{C}^{\mathrm{u}}$, and yet more, which can be achieved by simply lumping together all the language constructors and evaluation rules and proving that still "evaluation is invariant under change of notation". The separation of concerns suggested by singling out particular language features allows, however, for a more systematic language engineering by switching on and off certain features.

2.3. Variations in Evaluation

Up to here, the discussed OCL features all were evaluated over the same domain of OCL-closed order-sorted algebras. There are, however, also OCL constructs that necessitate extensions of this domain. Still, their definitions otherwise only involve term constructors, evaluation rules, and translations.

When it comes to `iterate` over arbitrary collections and not only sequences, OCL evaluation becomes *non-deterministic*. For this an extended domain of values $U^{\mathcal{P}}_{\Sigma}(M) = \mathcal{P}_{\Sigma}(S^M) = (\mathcal{P}_{\Sigma}(s^M))_{s\in|S|}$ for a Σ-algebra M has to be considered that involves powersets. The deterministic evaluation $[\![-]\!]^{\mathrm{it}}_{(\Sigma,X)}(M,\beta)\colon \mathscr{C}^{\mathrm{it}}_{\Sigma}(X) \to U^{\leq}_{\Sigma}(M)$ for a $\beta\colon X \to U^{\leq}_{\Sigma}(M)$ has to be lifted to a powerset-based evaluation function $[\![-]\!]^{\mathrm{nd}}_{(\Sigma,X)}(M,\beta^{\mathcal{P}})\colon \mathscr{C}^{\mathrm{it}}_{\Sigma}(X) \to U^{\mathcal{P}}_{\Sigma}(M)$ where $\beta^{\mathcal{P}}\colon X \to \mathcal{P}_{\Sigma}(X)$ is defined by $\beta^{\mathcal{P}}(x)_s = \{\beta(x)_s\}$. The composition $\xi^{\mathcal{P}}_2 \circ^{\mathcal{P}} \xi^{\mathcal{P}}_1\colon X_1 \to \mathcal{P}_{\Sigma}(X_3)$ of variable renamings $\xi^{\mathcal{P}}_1\colon X_1 \to \mathcal{P}_{\Sigma}(X_2)$ and $\xi^{\mathcal{P}}_2\colon X_2 \to \mathcal{P}_{\Sigma}(X_3)$ is given by $\{V \mapsto \bigcup_{v\in V} \xi^{\mathcal{P}}_2(v)\} \circ \xi^{\mathcal{P}}_1$. The clause for `iterate` over `Set(`s`)` involves a function $\mathit{Seq}_s\colon \mathcal{P}(s^M) \to \mathcal{P}((s^M)^*)$ yielding all possible sequences of values in a set, and reads

- $[\![c\texttt{->iterate(}i \texttt{ : } s\texttt{; } a \texttt{ : } s' \texttt{ = } e_0 \texttt{ | } e\texttt{)}]\!]^{\mathrm{nd}}_{(\Sigma,X)}(M,\beta^{\mathcal{P}})_{s'} =$
$$\begin{aligned} &\{\mathit{it}^M(v_c, v_{e_0}, h) \mid \\ &\quad v_c \in \mathit{Seq}_s(v_{\texttt{Set(}s\texttt{)}}), \\ &\qquad v_{\texttt{Set(}s\texttt{)}} \in [\![c]\!]^{\mathrm{nd}}_{(\Sigma,X)}(M,\beta^{\mathcal{P}})_{\texttt{Set(}s\texttt{)}}, \\ &\quad v_{e_0} \in [\![e_0]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\mathcal{P}})_{s'}, \\ &\quad h \in \{\{(v_i,v_a) \mapsto v_e\} \mid v_e \in [\![e]\!]^{\mathrm{nd}}_{(\Sigma,X\uplus\{i:s,a:s'\})} \\ &\qquad (M,\beta^{\mathcal{P}}\{i : s \mapsto \{v_i\}, a : s' \mapsto \{v_a\}\})_{s'}\}\,. \end{aligned}$$

Whereas expressions in previous versions of OCL precisely contained the primitive recursive functions (Cengarle & Knapp 2004), OCL 2.4 sports an unbounded *closure* construct

c`->closure(`i `:` s `|` e`)`

that computes the smallest set containing the set c, the e-successors of c, the e-successors of the e-successors of c etc., where an e-successor of a v is obtained by evaluating e with i bound to v. Thus, when including a special term formation rule for closures,

- $c\texttt{->closure(}i \texttt{ : } s \texttt{ | } e\texttt{)} \in \mathscr{C}^{\mathrm{c}}_{\Sigma}(X)_{\texttt{Set(}s\texttt{)}}$ for $s \in |S|$ where $c \in \mathscr{C}^{\mathrm{c}}_{\Sigma}(X)_{\texttt{Set(}s\texttt{)}}$ and $e \in \mathscr{C}^{\mathrm{c}}_{\Sigma}(X \uplus \{i : s\})_{\texttt{Set(}s\texttt{)}}$

the evaluation has to allow for possible *non-termination*, like in `Set{0}->closure(i : Int | Set{i+1})`. We can consider another extended domain of values $U^{\perp}_{\Sigma}(M) = \perp_{\Sigma}(S^M) = (s^M \uplus \{\perp\})_{s\in|S|}$ including $\perp$ for non-termination and use valuations of the form $\beta^{\perp}\colon X \to U^{\perp}_{\Sigma}(M)$ which are composed such that $\perp$ is preserved. Similarly to iterations we can then handle closures by an ad hoc higher-order function with unbounded accumulation $\mathit{cl}^M_{\perp}\colon (\mathcal{P}(s^M) \cup \{\perp\}) \times (s^M \to \mathcal{P}(s^M) \cup \{\perp\}) \times \mathcal{P}(s^M) \to \mathcal{P}(s^M) \cup \{\perp\}$:

- $[\![c\texttt{->closure(}i \texttt{ : } s \texttt{ | } e\texttt{)}]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\perp})_{\texttt{Set(}s\texttt{)}} =$
$$\begin{aligned} &\mathit{cl}^M_{\perp}([\![c]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\perp})_{\texttt{Set(}s\texttt{)}}, \\ &\qquad \{v_i \mapsto [\![e]\!]^{\mathrm{c}}_{(\Sigma,X\uplus\{i:s\})}(M,\beta^{\perp}\{i : s \mapsto v_i\})_{\texttt{Set(}s\texttt{)}}\}, \\ &\qquad \emptyset)\,, \end{aligned}$$
$$\begin{aligned} &\mathit{cl}^M_{\perp}(\emptyset, h, V) = V_{\perp}, \\ &\mathit{cl}^M(C, h, V) = \mathit{cl}^M((\textstyle\bigcup^{\perp}_{v\in C} h(v)) \setminus V, h, V \cup C), \end{aligned}$$

where $V_{\perp}$ is $\perp$ if V is infinite and V otherwise, $\bigcup^{\perp}$ yields $\perp$ if one of its summands is $\perp$, and $\mathit{cl}^M_{\perp}$ is strict in its first argument. Clearly, also the conditional has to be adapted and its evaluation considering non-termination reads:

- $[\![\texttt{if } e \texttt{ then } e_{\mathit{tt}} \texttt{ else } e_{\mathit{ff}} \texttt{ endif}]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\perp})_s =$
$$\begin{cases} [\![e_b]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\perp})_s & \text{if } [\![e]\!]^{\mathrm{c}}_{(\Sigma,X)}(M,\beta^{\perp})_{\texttt{Bool}} = b \\ & \quad \text{with } b \in \{\mathit{ff},\mathit{tt}\}\,, \\ \perp & \text{otherwise}\,. \end{cases}$$

3. Term Charter Domains and Term Charters

Having illustrated what is needed to formalise the semantics of OCL types and expressions to be evaluated over a single system state, we now provide a rather abstract account of all the ingredients used. On the one hand, the proposed framework (Knapp & Cengarle 2018) of *term charter domains* and *term charters* aims at making transparent and precise the required interactions of the various parts of signatures and translations, (algebraic) structures, variables and renamings, values, and evaluation. On the other hand, the abstract formulation offers means to combine different instantiations of the framework for modular language engineering (Knapp & Cengarle 2015). What is more, and as is discussed in the next sections, it can also be used to move from the evaluation of expressions over a single system state, as in OCL invariants, to the evaluation of relational expressions over two system states, as in OCL pre-/post-conditions.

Term charter domains capture the signatures, values and variables, structures, and the underlying values. The grammar of expression formation, variable renaming, variable embedding, and the evaluation proper is collected into a term charter over such a domain requiring conditions for embedding variables into expressions as well as the invariance of evaluation under renaming and translation. We use indexed categories (Tarlecki et al. 1991; Wolter & Martini 1997; Diaconescu 2008; Sannella & Tarlecki 2012) as a foundation for the abstract framework, since they provide a close link with the algebraic signatures as indexes also used in the model-theoretic framework of institutions (Sannella & Tarlecki 2012).

Term charter domains involve indexed categories and indexed functors: An *indexed category* N over an index category I is a functor $N\colon I^{\mathrm{op}} \to \mathrm{Cat}$. An *indexed functor* $F\colon M \dot{\to} N$ between the I-indexed categories M and N is given by a family of functors $(F_i\colon M(i) \to N(i))_{i \in |I|}$ such that $F_{i_2}; N(u) = M(u); F_{i_1}$ for each $u\colon i_1 \to i_2$ in I, i.e., F is a natural transformation from M to N.

Term charters use lax indexed functors, that subsume indexed functors, and lax indexed natural transformations: A *lax indexed functor* $F\colon M \ddot{\to} N$ between the I-indexed categories M and N is given by families of functors $(F_i\colon M(i) \to N(i))_{i \in |I|}$ and natural transformations $(F_u\colon M(u); F_{i_1} \dot{\to} F_{i_2}; N(u))_{u \in I(i_1, i_2)}$ with $F_{1_i} = 1_{F_i}$ such that $F_{u_1;u_2} = (M(u_2) \mathbin{{}^*\!\!;} F_{u_1}); (F_{u_2} \mathbin{{}^*\!\!;} N(u_1))$ for all $u_1 \in I(i_1, i_2)$, $u_2 \in I(i_2, i_3)$ — where $\mathbin{{}^*\!\!;}$ denotes the horizontal composition of natural transformations. The composition $F; G\colon L \ddot{\to} N$ of $F\colon L \ddot{\to} M$ and $G\colon M \ddot{\to} N$ for the I-indexed categories L, M, N is given by $(F; G)_i = F_i; G_i$ for $i \in |I|$ and $(F; G)_u = (F_u \mathbin{{}^*\!\!;} G_{i_1}); (F_{i_2} \mathbin{{}^*\!\!;} G_u)$ for $u \in I(i_1, i_2)$. For M, N I-indexed categories and $F, G\colon M \ddot{\to} N$ lax indexed functors, a *lax indexed natural transformation* $\eta\colon F \dot{\to} G$ is given by a family of natural transformations $(\eta_i\colon F_i \dot{\to} G_i)_{i \in |I|}$ such that $(M(u) \mathbin{{}^*\!\!;} \eta_{i_1}); G_u = F_u; (\eta_{i_2} \mathbin{{}^*\!\!;} N(u))$ for $u \in I(i_1, i_2)$.

3.1. Term Charter Domains

A *term charter domain* $(\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ is given by a category $\mathbb{S}$ of *signatures*, an indexed category $\mathit{Val}\colon \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$ of *value variables*, an indexed category $\mathit{Str}\colon \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$ of *structures*, and an *underlying* indexed functor $U\colon \mathit{Str} \dot{\to} \mathit{Val}$.

We use the terminology of "value variables", since Val plays the rôle of both. In fact, variables in Val can only be assigned values in Val, not, e.g., sets of values or functions.

Example. (1) A term charter domain $\mathsf{D}^{\le} = (\mathbb{S}^{\le}, \mathit{Val}^{\le}, \mathit{Str}^{\le}, U^{\le})$ for order-sorted algebras can be obtained by assembling the relevant parts of the previous section: The signature category $\mathbb{S}^{\le}$ comprises the order-sorted signatures and signature morphisms. For a $\Sigma \in |\mathbb{S}^{\le}|$, the category $\mathit{Val}^{\le}(\Sigma)$ of Σ-value variables comprises the Σ-variables with their renaming morphisms, and the category $\mathit{Str}^{\le}(\Sigma)$ of Σ-structures the Σ-algebras and their morphisms; a $\sigma\colon \Sigma \to \Sigma'$ in $\mathbb{S}^{\le}$ induces the σ-reduct functors $\mathit{Val}^{\le}(\sigma) = -|\sigma\colon \mathit{Val}^{\le}(\Sigma') \to \mathit{Val}^{\le}(\Sigma)$ and $\mathit{Str}^{\le}(\sigma) = -|\sigma\colon \mathit{Str}^{\le}(\Sigma') \to \mathit{Str}^{\le}(\Sigma)$. Finally, the underlying value variables and renamings are given by $U^{\le}_{\Sigma}(M) = (s^M)_{s \in S}$ and $U^{\le}_{\Sigma}(\mu\colon M_1 \to M_2) = (\mu_s)_{s \in |S|}$, such that indeed $U^{\le}_{\Sigma} \circ \mathit{Str}^{\le}(\sigma) = \mathit{Val}^{\le}(\sigma) \circ U^{\le}_{\Sigma'}$.
(2) For reflecting the type hierarchy and standard library of OCL, the term charter domain $\mathsf{D}^{\circ} = (\mathbb{S}^{\circ}, \mathit{Val}^{\circ}, \mathit{Str}^{\circ}, U^{\circ})$ is obtained by restricting $\mathsf{D}^{\le}$: $\mathbb{S}^{\circ}$ shows all signatures with predefined sorts `Bool`, `Real`, `Seq(`s`)` and function symbols `0.0`, `+`, etc., as well as all signature morphisms preserving these predefined symbols; Val° is only defined on $\mathbb{S}^{\circ}$; the structures in Str° are those with standard interpretations of the predefined symbols; and the underlying functor U° operates only on Str°.
(3) For the non-termination extension, the term charter domain $(\mathbb{S}^{\le}, \mathit{Val}^{\perp}, \mathit{Str}^{\le}, U^{\perp})$ is defined using the indexed endofunctor $\perp\colon \mathit{Val}^{\le} \dot{\to} \mathit{Val}^{\le}$ with $\perp_{\Sigma}(X) = (X_s \uplus \{\perp\})_{s \in |S|}$ and $\perp_{\Sigma}(\xi) = (\xi_s\{\perp \mapsto \perp\})_{s \in |S|}$ for a Kleisli construction (Borceux 1994): The objects of $\mathit{Val}^{\perp}(\Sigma)$ and $\mathit{Val}^{\le}(\Sigma)$ coincide, but a morphism $\xi^{\perp}\colon X_1 \to X_2$ of $\mathit{Val}^{\perp}(\Sigma)$ is given by the $\mathit{Val}^{\le}(\Sigma)$-morphism $\xi\colon X_1 \to \perp_{\Sigma}(X_2)$; the $\perp(\Sigma)$-identities $1_X\colon X \to X$ are the inclusions $\iota_X\colon X \to \perp_{\Sigma}(X)$, the composition $\xi_1^{\perp}; \xi_2^{\perp}$ of $\xi_1^{\perp}\colon X_1 \to X_2$ and $\xi_2^{\perp}\colon X_2 \to X_3$ is $(\xi_1; \xi_2\{\perp \mapsto \perp\})^{\perp}\colon X_1 \to X_3$. The underlying functor $U^{\perp}\colon \mathit{Str}^{\le} \dot{\to} \mathit{Val}^{\perp}$ is chosen as $U^{\perp}_{\Sigma}(M) = U^{\le}_{\Sigma}(M)$ and $U^{\perp}_{\Sigma}(\mu\colon M_1 \to M_2) = (U^{\le}_{\Sigma}(\mu); \iota_{U^{\le}_{\Sigma}(M_2)})^{\perp}$.

3.2. Term Charters

Let $(\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ be a term charter domain. Let $\mathscr{C}\colon \mathit{Val} \ddot{\to} \mathit{Val}$ be a lax indexed functor, where the functors $\mathscr{C}_{\Sigma}\colon \mathit{Val}(\Sigma) \to \mathit{Val}(\Sigma)$ for each $\Sigma \in |\mathbb{S}|$ *construct terms* and *rename terms* along value variable renamings, and the natural transformations $\mathscr{C}_{\sigma}\colon \mathit{Val}(\sigma); \mathscr{C}_{\Sigma} \dot{\to} \mathscr{C}_{\Sigma'}; \mathit{Val}(\sigma)$ for each $\sigma \in \mathbb{S}(\Sigma, \Sigma')$ *translate terms* along signature morphisms. Let furthermore $\nu\colon 1_{\mathit{Val}} \dot{\to} \mathscr{C}$ be a lax indexed natural transformation, where the natural transformations $\nu_{\Sigma}\colon 1_{\mathit{Val}(\Sigma)} \dot{\to} \mathscr{C}_{\Sigma}$ for each $\Sigma \in |\mathbb{S}|$ *embed value variables into terms*. Finally, for each $\Sigma \in |\mathbb{S}|$, $X \in |\mathit{Val}(\Sigma)|$, and $M \in |\mathit{Str}(\Sigma)|$, let

$$(\mathit{ext}_{\Sigma})^M_X\colon \mathit{Val}(\Sigma)(X, U_{\Sigma}(M)) \to \mathit{Val}(\Sigma)(\mathscr{C}_{\Sigma}(X), U_{\Sigma}(M))$$

be a function *extending* a value variable valuation β into a term valuation $(\mathit{ext}_{\Sigma})^M_X(\beta)$. Then $(\mathscr{C}, \nu, \mathit{ext})$ is a *term charter* over $(\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ if the following *variable condition* (V), *renaming condition* (R), and *translation condition* (T) are satisfied:

(V) $\nu_{\Sigma}(X); (\mathit{ext}_{\Sigma})^M_X(\beta) = \beta$

for all $\Sigma \in |\mathbb{S}|$ and $\beta \in \mathit{Val}(\Sigma)(X, U_{\Sigma}(M))$, i.e., the term valuation over a value variable valuation extends the value variable valuation;

(R) $\mathscr{C}_{\Sigma}(\xi); (\mathit{ext}_{\Sigma})^M_{X_2}(\beta) = (\mathit{ext}_{\Sigma})^M_{X_1}(\xi; \beta)$

for all $\Sigma \in |\mathbb{S}|$, $\xi \in \mathit{Val}(\Sigma)(X_1, X_2)$, and $\beta \in \mathit{Val}(\Sigma)(X_2, U_{\Sigma}(M))$, i.e., the "renaming lemma" holds; and

(T) $\mathscr{C}_{\sigma}(X'); \mathit{Val}(\sigma)((\mathit{ext}_{\Sigma'})^{M'}_{X'}(\beta')) =$
$\qquad (\mathit{ext}_{\Sigma})^{\mathit{Str}(\sigma)(M')}_{\mathit{Val}(\sigma)(X')}(\mathit{Val}(\sigma)(\beta'))$

for all $\sigma \in \mathbb{S}(\Sigma, \Sigma')$ and $\beta' \in \mathit{Val}(\Sigma')(X', U_{\Sigma'}(M'))$, i.e., the "translation lemma" holds.

This notion of term charters in fact directly translates the properties of concrete evaluation as discussed in the previous section into an abstract framework: Assume that for each $\Sigma \in |\mathbb{S}|$ there is a faithful functor $\mathcal{U}_\Sigma \colon \mathit{Val}(\Sigma) \to \mathrm{Set}$, i.e., each $\mathit{Val}(\Sigma)$ is a concrete category. Writing X for $\mathcal{U}_\Sigma(X)$ when $X \in |\mathit{Val}(\Sigma)|$, ξ for $\mathcal{U}_\Sigma(\xi)$ when $\xi \in \mathit{Val}(\Sigma)(X_1, X_2)$, $-|\sigma$ for both $\mathit{Str}(\sigma)(-)$ and $\mathit{Val}(\sigma)(-)$, $[\![t]\!]_{(\Sigma,X)}(M, \beta)$ for $(\mathit{ext}_\Sigma)^M_X(\beta)(t)$, $\xi(t)$ for $\mathscr{T}_\Sigma(\xi)(t)$, and $\sigma(t)$ for $\mathscr{T}_\sigma(X')(t)$, the conditions become

(V) $[\![x]\!]_{(\Sigma,X)}(M, \beta) = \beta(x)$;
(R) $[\![\xi(t)]\!]_{(\Sigma,X_2)}(M, \beta) = [\![t]\!]_{(\Sigma,X_1)}(M, \beta \circ \xi)$;
(T) $([\![\sigma(t)]\!]_{(\Sigma',X')}(M', \beta'))|\sigma = [\![t]\!]_{(\Sigma,X'|\sigma)}(M'|\sigma, \beta'|\sigma)$.

Example. (1) For order-sorted algebras, $\mathfrak{T}^{\leq} = (\mathscr{T}^{\leq}, \nu^{\leq}, \mathit{ext}^{\leq})$ with $\nu^{\leq}_\Sigma(X) = 1_X \colon X \hookrightarrow \mathscr{T}^{\leq}_\Sigma(X)$ and $(\mathit{ext}^{\leq}_\Sigma)^M_X(\beta) = [\![-]\!]^{\leq}_{(\Sigma,X)}(M, \beta)$ forms a $(\mathbb{S}^{\leq}, \mathit{Val}^{\leq}, \mathit{Str}^{\leq}, U^{\leq})$-term charter. Note that, indeed, $\mathscr{T}^{\leq} \colon \mathit{Val}^{\leq} \mathrel{\dot{\to}} \mathit{Val}^{\leq}$.

(2) For iterations, $\mathfrak{T}^{\mathrm{it}} = (\mathscr{C}^{\mathrm{it}}, \nu^{\mathrm{it}}, \mathit{ext}^{\mathrm{it}})$ with $\nu^{\mathrm{it}}_\Sigma(X) = 1_X \colon X \hookrightarrow \mathscr{C}^{\mathrm{it}}_\Sigma(X)$, and $(\mathit{ext}^{\mathrm{it}}_\Sigma)^M_X(\beta) = [\![-]\!]^{\mathrm{it}}_{(\Sigma,X)}(M, \beta)$ constitutes a D°-term charter, as the instantiated properties (V_{it}), (R_{it}), and (T_{it}) hold by induction on term construction.

Similarly, the constructions for `allInstances` and the handling of invalid values by `isInvalid` and a non-strict `if` … `then` … `else` … `endif` can be comprised in D°-term charters $\mathfrak{T}^{\mathrm{a}} = (\mathscr{C}^{\mathrm{a}}, \nu^{\mathrm{a}}, \mathit{ext}^{\mathrm{a}})$ and $\mathfrak{T}^{\mathrm{u}} = (\mathscr{C}^{\mathrm{u}}, \nu^{\mathrm{u}}, \mathit{ext}^{\mathrm{u}})$ respectively.

Term charters over a term charter domain $\mathrm{D} = (\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ form the category $\mathrm{TmCh}(\mathrm{D})$ with a *term charter* (forward) *morphism* $\mu \colon \mathfrak{T}_1 \to \mathfrak{T}_2$ from $\mathfrak{T}_1 = (\mathscr{C}_1, \nu_1, \mathit{ext}_1)$ to $\mathfrak{T}_2 = (\mathscr{C}_2, \nu_2, \mathit{ext}_2)$ given by a lax indexed natural transformation $\mu \colon \mathscr{C}_1 \mathrel{\dot{\to}} \mathscr{C}_2$ such that both $\nu_{1,\Sigma}(X); \mu_\Sigma(X) = \nu_{2,\Sigma}(X)$ and $\mu_\Sigma(X); (\mathit{ext}_{2,\Sigma})^M_X(\beta) = (\mathit{ext}_{1,\Sigma})^M_X(\beta)$ hold for all $\Sigma \in |\mathbb{S}|$, $X \in |\mathit{Val}(\Sigma)|$, $M \in |\mathit{Str}(\Sigma)|$, and $\beta \colon X \to U_\Sigma(M)$; see (Knapp & Cengarle 2015).

3.3. Combining Term Charters

One of the main goals of the term charter framework is to support modular language development. We briefly discuss two combinators for term charters in $\mathrm{TmCh}(\mathrm{D})$ over the same term charter domain D; see (Knapp & Cengarle 2015). The first construction composes two term charters sequentially, the second one builds the "union" of term charters as a general co-limit construction. Together, these methods of composition allow for a deeply nested language combination.

Let $\mathfrak{T}_1 = (\mathscr{C}_1, \nu_1, \mathit{ext}_1)$ and $\mathfrak{T}_2 = (\mathscr{C}_2, \nu_2, \mathit{ext}_2)$ be term charters over the term charter domain $\mathrm{D} = (\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$. Then the *sequencing* $\mathfrak{T}_1 \rhd \mathfrak{T}_2 = (\mathscr{C}, \nu, \mathit{ext})$ of first $\mathfrak{T}_1$ and then $\mathfrak{T}_2$ is defined by $\mathscr{C} = \mathscr{C}_1; \mathscr{C}_2 \colon \mathit{Val} \mathrel{\dot{\to}} \mathit{Val}$, $\nu = \nu_1; (\mathscr{C}_1{}^*\nu_2) \colon 1_{\mathit{Val}} \mathrel{\dot{\to}} \mathscr{C}$, and

$$(\mathit{ext}_\Sigma)^M_X(\beta) = (\mathit{ext}_{2,\Sigma})^M_{\mathscr{C}_{1,\Sigma}(X)}((\mathit{ext}_{1,\Sigma})^M_X(\beta))$$

for all $\Sigma \in |\mathbb{S}|$, $X \in |\mathit{Val}(\Sigma)|$, $M \in |\mathit{Str}(\Sigma)|$, and $\beta \colon X \to U_\Sigma(M)$. In particular, $\mathfrak{T}_1 \rhd \mathfrak{T}_2$ is a term charter over D; see (Knapp & Cengarle 2015).

Example. Over the term charter domain D°, consider the expression

```
(Seq(Bool).allInstances()).isInvalid() .
```

This "heterogeneous" term in $\mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}}$ is built by first constructing `Seq(Bool).allInstances()` in $\mathfrak{T}^{\mathrm{a}} = (\mathscr{C}^{\mathrm{a}}, \nu^{\mathrm{a}}, \mathit{ext}^{\mathrm{a}})$, then taking this term as a variable, which we may abbreviate by $x \in \mathscr{C}^{\mathrm{a}}_\Sigma(X)$, and constructing `x.isInvalid()` in $\mathfrak{T}^{\mathrm{u}} = (\mathscr{C}^{\mathrm{u}}, \nu^{\mathrm{u}}, \mathit{ext}^{\mathrm{u}})$. Consequently, for an arbitrary valuation $\beta \colon X \to U^\circ_\Sigma(M)$ with value variables $X \in |\mathit{Val}^\circ(\Sigma)|$, structure $M \in |\mathit{Str}^\circ(\Sigma)|$, and signature $\Sigma \in |\mathbb{S}^\circ|$, to evaluate

$$\begin{aligned}&(\mathit{ext}^{\mathrm{u}}_\Sigma)^M_{\mathscr{C}^{\mathrm{a}}_\Sigma(X)}((\mathit{ext}^{\mathrm{a}}_\Sigma)^M_X(\beta))\\ &\quad(\texttt{(Seq(Bool).allInstances()).isInvalid()}) =\\ &(\mathit{ext}^{\mathrm{u}}_\Sigma)^M_{\mathscr{C}^{\mathrm{a}}_\Sigma(X)}((\mathit{ext}^{\mathrm{a}}_\Sigma)^M_X(\beta))(\texttt{x.isInvalid()})\end{aligned}$$

consists, firstly, in evaluating $(\mathit{ext}^{\mathrm{u}}_\Sigma)^M_{\mathscr{C}^{\mathrm{a}}_\Sigma(X)}((\mathit{ext}^{\mathrm{a}}_\Sigma)^M_X(\beta))(x)$, which amounts to $(\mathit{ext}^{\mathrm{a}}_\Sigma)^M_X(\beta)(x)$, since x is a variable in $\mathscr{C}^{\mathrm{a}}_\Sigma(X)$. Secondly, given that $\{0, \mathit{ff}, \mathit{tt}\}^*$ is an infinite set, $(\mathit{ext}^{\mathrm{a}}_\Sigma)^M_X(\beta)(\texttt{Seq(Bool).allInstances()})$ evaluates to $\dagger$. Thus the overall result is tt.

Also the lax indexed natural transformation $\mathscr{C}_1{}^*\nu_2 \colon \mathscr{C}_1 \mathrel{\dot{\to}} \mathscr{C}_1; \mathscr{C}_2$ induces a term charter morphism from $\mathfrak{T}_1$ to $\mathfrak{T}_1 \rhd \mathfrak{T}_2$, and, likewise, the natural transformation $\nu_1{}^*\mathscr{C}_2 \colon \mathscr{C}_2 \mathrel{\dot{\to}} \mathscr{C}_1; \mathscr{C}_2$ induces a term charter morphism from $\mathfrak{T}_2$ to $\mathfrak{T}_1 \rhd \mathfrak{T}_2$. The *$n$-th iteration* $\mathfrak{T}^{\rhd n}$ of a term charter $\mathfrak{T}$ for $n \geq 1$ is inductively defined by $\mathfrak{T}^{\rhd 1} = \mathfrak{T}$ and $\mathfrak{T}^{\rhd(n+1)} = \mathfrak{T}^{\rhd n} \rhd \mathfrak{T}$.

The second method of composition is building a co-limit for a diagram $F \colon J \to \mathrm{TmCh}(\mathrm{D})$ where J is a small connected category. Writing $\mathfrak{T}_{\mathscr{C}}$ for the first component of a term charter $\mathfrak{T} = (\mathscr{C}, \nu, \mathit{ext}) \in |\mathrm{TmCh}(\mathrm{D})|$, assume that all the diagrams $F_{\mathscr{C},X} \colon J \to \mathit{Val}(\Sigma)$ with $F_{\mathscr{C},X}(j) = (F(j)_{\mathscr{C}})_\Sigma(X)$ and $F_{\mathscr{C},X}(f \colon j \to j') = F(f)_\Sigma(X)$ for $\Sigma \in |\mathbb{S}|$, $X \in |\mathit{Val}(\Sigma)|$ have co-limits. From the universality of co-limits it follows that a term charter $\mathfrak{T}^F = (\mathscr{C}^F, \nu^F, \mathit{ext}^F)$ can be defined uniquely (up to isomorphism), where $\mathscr{C}^F \colon \mathit{Val} \mathrel{\dot{\to}} \mathit{Val}$ just maps $\mathscr{C}_\Sigma(X)$ to the co-limit object of $F_{\mathscr{C},X}$; see (Knapp & Cengarle 2015).

Example. As in the previous example, consider the D°-term charter $\mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}}$. We now want to construct arbitrarily nested terms from $\mathfrak{T}^{\mathrm{a}}$ and $\mathfrak{T}^{\mathrm{u}}$. We thus consider the chain $\mathfrak{T} \xrightarrow{\nu_1} \mathfrak{T}^{\rhd 2} \xrightarrow{\nu_2} \mathfrak{T}^{\rhd 3} \xrightarrow{\nu_3} \cdots$ for $\mathfrak{T} = \mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}}$. Writing $\mathscr{C}$ for the construction functor $\mathscr{C}^{\mathrm{a}}; \mathscr{C}^{\mathrm{u}}$, we have to check that the chain $\mathscr{C}_\Sigma(X) \xrightarrow{\nu_{1,\Sigma}(X)} \mathscr{C}^{\rhd 2}_\Sigma(X) \xrightarrow{\nu_{2,\Sigma}(X)} \mathscr{C}^{\rhd 3}_\Sigma(X) \xrightarrow{\nu_{3,\Sigma}(X)} \cdots$ has a co-limit in $\mathit{Val}^\circ(\Sigma)$ for $X \in |\mathit{Val}^\circ(\Sigma)|$. Indeed, the co-limit object of this chain is simply given by the component-wise union of the value domains. We obtain a co-limit term charter, which we denote by $\mathfrak{T}^{\rhd *}$. The evaluation of a term in $(\mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}})^{\rhd *}$ at a nesting level n, then, proceeds like in $(\mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}})^{\rhd n}$.

4. OCL Contracts

OCL expressions are evaluated over a single system state and thus can be used to query and navigate snapshots of a system or constrain system states by invariants. The evaluation of such expressions in states is abstractly captured by term charters over term charter domains. Besides expressions for invariant constraints, OCL offers pre-/post-conditions for operation contracts, like the following for describing the withdrawal of an amount from an account in our example from Fig. 1:

```
context Account::withdraw(amount : Real)
pre:  amount > 0 and self.balance >= amount
post: self.balance = self.balance@pre - amount
```

In contrast to the term charters above, now two structures have to be involved, the first describing *pre-condition time*, i.e., the state before `withdraw` is performed and to which attributes adorned with `@pre` refer, the second *post-condition time*, i.e., after `withdraw` has been performed and to which the unadorned attributes refer. On the other hand, the variables `self` and `amount` have only to be bound to a value once and must not change from pre-condition to post-condition time. We describe an approach where two structures can be combined into a single one such that the following formula φ_{wd}

```
    amount > 0 and self.balance@pre >= amount
and self.balance = self.balance@pre - amount
```

can be evaluated over the variables `self` and `amount` by the same means of term charters, but over an extended signature, technically constructed by a push-out, and a combined structure, using a weak form of amalgamation.

The main aim is to derive the requirements on term charter domains and term charters in order to make them "2-able", i.e., to see them prepared for a lifting from a single system state to two system states for pre- and post-condition time, which technically means to ensure that there are push-outs and a weak form of amalgamation. We use OCL as rôle model and the accounts example in Fig. 1, based on the term charter $(\mathscr{T}^{\le}, \nu^{\le}, \mathit{ext}^{\le})$ over the term charter domain $(\mathbb{S}^{\le}, \mathit{Val}^{\le}, \mathit{Str}^{\le}, \mathit{U}^{\le})$, as primary illustration; for simplicity, however, we omit SavingsAccount from our discussion.

Let $\mathrm{Ac}_{\mathrm{b}} = (\mathrm{S}_{\mathrm{b}}, \mathrm{F}_{\mathrm{b}})$ be a *basic* order-sorted signature in $\mathbb{S}^{\le}$ comprising the OCL sorts `Bool` and `Real` and showing function declarations for the real constant `0`: `Real`, the subtraction of reals `-`: `Real` $\times$ `Real` $\rightarrow$ `Real`, and the comparison operators on reals `>`, `>=`, `=`: `Real` $\times$ `Real` $\rightarrow$ `Bool`. Now consider a *sort extension* $\mathrm{Ac}_{\mathrm{c}} = (\mathrm{S}_{\mathrm{c}}, \mathrm{F}_{\mathrm{c}})$ of Ac_{b} adding the sort Account such that $\sigma_{\mathrm{c}} \colon \mathrm{Ac}_{\mathrm{b}} \hookrightarrow \mathrm{Ac}_{\mathrm{c}}$ is an inclusion of order-sorted signatures (i.e., $\sigma_{\mathrm{c},\mathrm{S}}(s) = s$ for all $s \in |\mathrm{S}_{\mathrm{c}}|$ and $\sigma_{\mathrm{c},\mathrm{F}}(f) = f$ for each $f \in \mathrm{F}_{\mathrm{c},\bar{s}}$ with $\bar{s} \in |\mathrm{S}_{\mathrm{c}}|^{+}$). Furthermore, consider two *declaration extensions* of Ac_{c}: On the one hand, $\mathrm{Ac}_{\mathrm{cd}}$ adds a declaration for balance: Account $\rightarrow$ `Real`, such that $\sigma_{\mathrm{cd}} \colon \mathrm{Ac}_{\mathrm{c}} \hookrightarrow \mathrm{Ac}_{\mathrm{cd}}$ is an inclusion of order-sorted signatures; on the other hand, the *renamed* version $\overleftarrow{\mathrm{Ac}}_{\mathrm{cd}}$ of this extension replaces balance by balance`@pre` yielding the accompanying order-sorted signature inclusion morphism $\overleftarrow{\sigma}_{\mathrm{cd}} \colon \mathrm{Ac}_{\mathrm{c}} \hookrightarrow \overleftarrow{\mathrm{Ac}}_{\mathrm{cd}}$. Let now $\widehat{\mathrm{Ac}}$ with $\widehat{\overleftarrow{\sigma}} \colon \overleftarrow{\mathrm{Ac}}_{\mathrm{cd}} \hookrightarrow \widehat{\mathrm{Ac}}$ and $\widehat{\sigma} \colon \mathrm{Ac}_{\mathrm{cd}} \hookrightarrow \widehat{\mathrm{Ac}}$ be the push-out of Ac_{c} along $\overleftarrow{\sigma}_{\mathrm{cd}}$ and σ_{cd} resulting in the following commuting diagram of order-sorted signatures in $\mathbb{S}^{\le}$:

$$\begin{tikzcd}
 & & \overleftarrow{\mathrm{Ac}}_{\mathrm{cd}} \arrow[dr, hook, "\widehat{\overleftarrow{\sigma}}"] & \\
\mathrm{Ac}_{\mathrm{b}} \arrow[r, hook, "\sigma_{\mathrm{c}}"] & \mathrm{Ac}_{\mathrm{c}} \arrow[ur, hook, "\overleftarrow{\sigma}_{\mathrm{cd}}"] \arrow[dr, hook, "\sigma_{\mathrm{cd}}"'] & & \widehat{\mathrm{Ac}} \\
 & & \mathrm{Ac}_{\mathrm{cd}} \arrow[ur, hook, "\widehat{\sigma}"'] &
\end{tikzcd}$$

More generally and abstractly, given a category $\mathbb{S}$, $(\Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2, (\hat{\sigma}_i \colon \Sigma_i \rightarrow \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2)_{1 \le i \le 2})$ is a *push-out* along the morphisms $\sigma_1 \colon \Sigma_0 \rightarrow \Sigma_1$ and $\sigma_2 \colon \Sigma_0 \rightarrow \Sigma_2$ in $\mathbb{S}$ if $\sigma_1 ; \hat{\sigma}_1 = \sigma_2 ; \hat{\sigma}$ and, furthermore, for all $\Sigma' \in |\mathbb{S}|$ and $\sigma_i' \colon \Sigma_i \rightarrow \Sigma'$, $i \in \{1, 2\}$ satisfying $\sigma_1 ; \sigma_1' = \sigma_2 ; \sigma_2'$, there is a unique morphism $\sigma \colon \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2 \rightarrow \Sigma'$ with $\sigma_i' = \hat{\sigma}_i ; \sigma$, $i \in \{1, 2\}$ as shown in following commuting diagram:

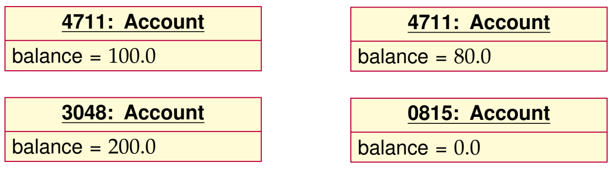
$$\begin{tikzcd}
 & \Sigma_1 \arrow[dr, "\hat{\sigma}_1"] \arrow[drr, bend left, "\sigma_1'"] & & \\
\Sigma_0 \arrow[ur, "\sigma_1"] \arrow[dr, "\sigma_2"'] & & \Sigma_1 +_{\Sigma_0}^{\sigma_1,\sigma_2} \Sigma_2 \arrow[r, dashed, "\sigma"] & \Sigma' \\
 & \Sigma_2 \arrow[ur, "\hat{\sigma}_2"'] \arrow[urr, bend right, "\sigma_2'"'] & &
\end{tikzcd}$$

Thus the push-out object $\widehat{\mathrm{Ac}} = \overleftarrow{\mathrm{Ac}}_{\mathrm{cd}} +_{\mathrm{Ac}_{\mathrm{b}}}^{\sigma_{\mathrm{c}};\overleftarrow{\sigma}_{\mathrm{cd}}, \sigma_{\mathrm{c}};\sigma_{\mathrm{cd}}} \mathrm{Ac}_{\mathrm{cd}}$ (which exists in order-sorted algebras) shows all parts of Ac_{b} as well as a single sort Account and two function declarations for balance and balance`@pre`. Note that, if the sort extension is not separated from the declaration extension, that is, if these two extensions are done in a single step, the push-out object would also duplicate the sorts. Note also that it is technically not necessary to first come up with a renamed copy of $\mathrm{Ac}_{\mathrm{cd}}$, which corresponds to Accs of Sect. 2.1, as the push-out construction takes care of providing two separate copies of all those features of $\mathrm{Ac}_{\mathrm{cd}}$ that are not shared in Ac_{c}. Still, preparing a designated copy allows to use pre-formed names like …`@pre`.

Now consider again the conjunctive interpretation φ_{wd} of the pre-/post-condition pair for the withdrawal contract. With our prerequisites on signatures, it forms an expression in $\mathscr{T}^{\le}_{\widehat{\mathrm{Ac}}}(\widehat{\mathrm{X}})$ over the combined signature $\widehat{\mathrm{Ac}}$ and the $\widehat{\mathrm{Ac}}$-value variables $\widehat{\mathrm{X}}$ comprising `self : `Account and `amount : Real`. In fact, the value variables $\widehat{\mathrm{X}}$ could also reflect that some variables are present already at pre-condition time, as in our case both `self` and `amount`, but others, like `result` for storing the value returned by an operation call, only at post-condition time. This can be achieved by combining value variables $\overleftarrow{\mathrm{X}}_{\mathrm{cd}} \in |\mathit{Val}^{\le}(\overleftarrow{\mathrm{Ac}}_{\mathrm{cd}})|$ and $\mathrm{X}_{\mathrm{cd}} \in |\mathit{Val}^{\le}(\mathrm{Ac}_{\mathrm{cd}})|$ into their disjoint union $\widehat{\mathrm{X}} = \overleftarrow{\mathrm{X}}_{\mathrm{cd}} \uplus \mathrm{X}_{\mathrm{cd}} \in |\mathit{Val}^{\le}(\widehat{\mathrm{Ac}})|$; the same holds for valuations that go from (value) variables to value (variable)s.

Next, let $\mathrm{St}_{\mathrm{b}} = (\mathrm{S}_{\mathrm{b}}^{\mathrm{St}_{\mathrm{b}}}, \mathrm{F}_{\mathrm{b}}^{\mathrm{St}_{\mathrm{b}}})$ be a canonical order-sorted structure in $\mathit{Str}^{\le}(\mathrm{Ac}_{\mathrm{b}})$ interpreting the sorts `Bool` and `Real` as well as the function declarations for `0`, `-`, `>`, `>=`, and `=` as usual. Also let $\overleftarrow{\mathrm{St}}_{\mathrm{cd}} = (\mathrm{S}_{\mathrm{c}}^{\overleftarrow{\mathrm{St}}_{\mathrm{cd}}}, \overleftarrow{\mathrm{F}}_{\mathrm{cd}}^{\overleftarrow{\mathrm{St}}_{\mathrm{cd}}}) \in |\mathit{Str}^{\le}(\overleftarrow{\mathrm{Ac}}_{\mathrm{cd}})|$ be an order-sorted algebra representing a system state at pre-condition time and $\mathrm{St}_{\mathrm{cd}} = (\mathrm{S}_{\mathrm{c}}^{\mathrm{St}_{\mathrm{cd}}}, \mathrm{F}_{\mathrm{cd}}^{\mathrm{St}_{\mathrm{cd}}}) \in |\mathit{Str}^{\le}(\mathrm{Ac}_{\mathrm{cd}})|$ an order-sorted algebra for a post-state such that the canonical structure St_{b} is shared by both, i.e.,

$$\mathit{Str}^{\le}(\overleftarrow{\sigma}_{\mathrm{cd}} \circ \sigma_{\mathrm{c}})(\overleftarrow{\mathrm{St}}_{\mathrm{cd}}) \cong \mathrm{St}_{\mathrm{b}} \cong \mathit{Str}^{\le}(\sigma_{\mathrm{cd}} \circ \sigma_{\mathrm{c}})(\mathrm{St}_{\mathrm{cd}}) \ .$$



(a) Pre-state $\overleftarrow{\mathrm{St}}_{\mathrm{cd}}$ **(b)** Post-state $\mathrm{St}_{\mathrm{cd}}$

Figure 2 UML object diagrams for accounts

Let these algebras show interpretations of Account and balance resp. balance@pre as follows (cf. the UML object diagrams in Fig. 2):

$$
\begin{array}{ll}
S^{\overleftarrow{\mathrm{St}}_{\mathrm{cd}}}_{\mathrm{c,Account}} = \{\mathtt{3048}, \mathtt{4711}\} & S^{\mathrm{St}_{\mathrm{cd}}}_{\mathrm{c},\mathtt{Account}} = \{\mathtt{4711}, \mathtt{0815}\} \\
\mathsf{balance@pre}^{\overleftarrow{\mathrm{St}}_{\mathrm{cd}}}(\mathtt{4711}) = 100.0 & \mathsf{balance}^{\mathrm{St}_{\mathrm{cd}}}(\mathtt{4711}) = 80.0 \\
\mathsf{balance@pre}^{\overleftarrow{\mathrm{St}}_{\mathrm{cd}}}(\mathtt{3048}) = 200.0 & \mathsf{balance}^{\mathrm{St}_{\mathrm{cd}}}(\mathtt{0815}) = 0.0
\end{array}
$$

In this setting, at pre-condition time, i.e., in $\overleftarrow{\mathrm{St}}_{\mathrm{cd}}$, there is an account 3048 with balance 200.0 and an account 4711 with balance 100.0. At post-condition time, i.e., in $\mathrm{St}_{\mathrm{cd}}$, there still is the account 4711, now with balance 80.0, additionally an account 0815 with balance 0.0, and account 3048 has been closed.

We may now form $\widehat{\mathrm{St}} = (\widehat{\mathrm{S}}^{\widehat{\mathrm{St}}}, \widehat{\mathrm{F}}^{\widehat{\mathrm{St}}}) \in |\mathit{Str}^{\leq}(\widehat{\mathrm{Ac}})|$ by taking the "union" of $\overleftarrow{\mathrm{St}}_{\mathrm{cd}}$ and $\mathrm{St}_{\mathrm{cd}}$ as combined order-sorted algebra over the combined signature $\widehat{\mathrm{Ac}} = (\widehat{\mathrm{S}}, \widehat{\mathrm{F}})$: For all basic sorts and function declarations we choose $\widehat{\mathrm{St}}$ to behave like St_{b}, and for the additional sorts and function declarations contributed in Ac_{c}, $\overleftarrow{\mathrm{Ac}}_{\mathrm{cd}}$, and $\mathrm{Ac}_{\mathrm{cd}}$, we set

$$
\begin{array}{ll}
\widehat{S}^{\widehat{\mathrm{St}}}_{\mathtt{Account}} = \{\mathtt{3048}, \mathtt{4711}, \mathtt{0815}\} & \\
\mathsf{balance@pre}^{\widehat{\mathrm{St}}}(\mathtt{3048}) = 200.0 & \mathsf{balance}^{\widehat{\mathrm{St}}}(\mathtt{3048}) = 0.0 \\
\mathsf{balance@pre}^{\widehat{\mathrm{St}}}(\mathtt{0815}) = 0.0 & \mathsf{balance}^{\widehat{\mathrm{St}}}(\mathtt{0815}) = 0.0 \\
\mathsf{balance@pre}^{\widehat{\mathrm{St}}}(\mathtt{4711}) = 100.0 & \mathsf{balance}^{\widehat{\mathrm{St}}}(\mathtt{4711}) = 80.0
\end{array}
$$

In particular, $\mathit{Str}^{\leq}(\widehat{\sigma} \circ \sigma_{\mathrm{cd}} \circ \sigma_{\mathrm{c}})(\widehat{\mathrm{St}}) \cong \mathrm{St}_{\mathrm{b}}$, but it deserves separate mention that $\mathit{Str}^{\leq}(\widehat{\sigma})(\widehat{\mathrm{St}}) \ncong \mathrm{St}_{\mathrm{cd}}$ as their values differ. In fact, there is considerable freedom how to "construct" the combined structure $\widehat{\mathrm{St}}$ from $\overleftarrow{\mathrm{St}}_{\mathrm{cd}}$ and $\mathrm{St}_{\mathrm{cd}}$. Here, we have used the "default value" 0.0 for the missing pieces of information for $\mathsf{balance@pre}^{\widehat{\mathrm{St}}}(\mathtt{0815})$ and $\mathsf{balance}^{\widehat{\mathrm{St}}}(\mathtt{3048})$, but the choice is only limited by what is available in the underlying structures, like 0 in structures with an explicit value for "null" (which is prescribed by OCL and can be achieved by using D°) or just leaving the definition open in partial algebras.

More generally and abstractly, we say that an indexed category $\mathit{Str}\colon \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$ has *weak union amalgamations* w.r.t. to a subcategory $\mathbb{S}^*$ of $\mathbb{S}$ that has push-outs, if for all $\sigma_i\colon \Sigma_0 \to \Sigma_i$ in $\mathbb{S}^*$, $i \in \{1,2\}$,

- for all $M_i \in |\mathit{Str}(\Sigma_i)|$, $i \in \{1,2,3\}$, with $\mathit{Str}(\sigma_1)(M_1) \cong M_0 \cong \mathit{Str}(\sigma_2)(M_2)$, there is a structure $M_1 \times_{\Sigma_0} M_2 \in |\mathit{Str}(\Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2)|$ such that $\mathit{Str}(\sigma_1;\hat{\sigma}_1)(M_1 \times_{\Sigma_0} M_2) \cong M_0 \cong \mathit{Str}(\sigma_2;\hat{\sigma}_2)(M_1 \times_{\Sigma_0} M_2)$;
- for all $\mu_i\colon M_i \to N_i$ in $|\mathit{Str}(\Sigma_i)|$, $i \in \{1,2,3\}$, with $\mathit{Str}(\sigma_1)(\mu_1) \cong \mu_0 \cong \mathit{Str}(\sigma_2)(\mu_2)$, there is a structure morphism $\mu_1 \times_{\Sigma_0} \mu_2\colon M_1 \times_{\Sigma_0} M_2 \to N_1 \times_{\Sigma_0} N_2$ in $\mathit{Str}(\Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2)$ such that $\mathit{Str}(\sigma_1;\hat{\sigma}_1)(\mu_1 \times_{\Sigma_0} \mu_2) \cong \mu_0 \cong \mathit{Str}(\sigma_2;\hat{\sigma}_2)(\mu_1 \times_{\Sigma_0} \mu_2)$;
- for all signature morphisms $\rho_i\colon \Sigma_i \to \Sigma'_i$ in $\mathbb{S}^*$, $0 \leq i \leq 2$, and the unique signature morphism $\hat{\rho}\colon \Sigma_1 +^{\sigma_1,\sigma_2}_{\Sigma_0} \Sigma_2 \to \Sigma'_1 +^{\sigma'_1,\sigma'_2}_{\Sigma'_0} \Sigma'_2$, it holds that $\mathit{Str}(\hat{\rho})(M'_1 \times_{\Sigma'_0} M'_2) = \mathit{Str}(\rho_1)(M'_1) \times_{\Sigma_0} \mathit{Str}(\rho_2)(M'_2)$.

In our example, the formation of $\widehat{\mathrm{St}} = \overleftarrow{\mathrm{St}}_{\mathrm{cd}} \times_{\mathrm{Ac_b}} \mathrm{St}_{\mathrm{cd}}$ corresponds to the "union" described above. It also satisfies the requirement that reducts of weak union amalgamations are equal to weak union amalgamations of reducts, which ensures compatibility with translations. Equality of structures can indeed be weakened to an equality of the underlying values.

Now reconsider $\varphi_{\mathrm{wd}} \in \mathscr{T}^{\leq}_{\widehat{\mathrm{Ac}}}(\widehat{\mathrm{X}})$ from the pre-/post-condition pair for the withdrawal contract with the $\widehat{\mathrm{Ac}}$-value variables $\widehat{\mathrm{X}}$ comprising `self : Account` and `amount : Real`. Using the combined structure $\widehat{\mathrm{St}}$, the expression spanning two states can be evaluated using

$$(\mathit{ext}^{\leq}_{\widehat{\mathrm{Ac}}})^{\widehat{\mathrm{St}}}_{\widehat{\mathrm{X}}}(\hat{\beta}) = [\![-]\!]^{\leq}_{(\widehat{\mathrm{Ac}},\widehat{\mathrm{X}})}(\widehat{\mathrm{St}}, \hat{\beta})$$

for a value variable assignment $\hat{\beta}\colon \widehat{X} \to U^{\leq}_{\widehat{\mathrm{Ac}}}(\widehat{\mathrm{St}})$ given by, say, $\hat{\beta}_{\mathsf{Account}}(\mathtt{self}) = \mathtt{4711}$ and $\hat{\beta}_{\mathtt{Real}}(\mathtt{amount}) = 20.0$. Then $(\mathit{ext}^{\leq}_{\widehat{\mathrm{Ac}}})^{\widehat{\mathrm{St}}}_{\widehat{\mathrm{X}}}(\hat{\beta})_{\mathtt{Bool}}(\varphi_{\mathrm{wd}}) = \mathit{tt}$, i.e., the withdrawal contract is satisfied for this particular situation. By contrast, when choosing $\hat{\beta}_{\mathsf{Account}}(\mathtt{self}) = \mathtt{3048}$, the result is ff.

Finally, and more generally and abstractly, we call a term charter domain $\mathrm{D} = (\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ *2-able* if there is a subcategory $\mathbb{S}^*$ of $\mathbb{S}$ with push-outs and Str has weak union amalgamations as of above w.r.t. $\mathbb{S}^*$.

It has to be noted that `allInstances` is not properly supported by this approach: The evaluation of the expression `Account.allInstances()` would return the accounts available at both pre-condition and post-condition time which is not what OCL requires. Moreover, OCL's `isNew()`, which tests whether some object did not exist at the pre-state, needs special attention.

5. 2-Term Domains and 2-Term Charters

In order to handle OCL contracts abstractly we move from a 2-able term charter domain D to a 2-term charter domain and from a given term charter $\mathfrak{T}$ over D to a 2-term charter. The former has as 2-signatures extensions of a basic D-signature by first "sorts" and then "declarations", and as 2-structures two D-structures for a pre-state and a post-state. The latter constructs expressions over the push-out signature in D and uses the evaluation from $\mathfrak{T}$ for the weak union amalgamation of the pre-state and the post-state. However, 2-term charter domains and 2-term charters are just special cases of term charter domains and term charters respectively.

5.1. 2-Term Charter Domains

Let $\mathsf{D} = (\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ be a 2-able term charter domain. Let $\mathbb{S}^*$ be a subcategory of $\mathbb{S}$ showing push-outs and $\times$ a weak union amalgamation construction for Str w.r.t. $\mathbb{S}^*$. The *2-term charter domain* $2\mathsf{D}^* = (2\mathbb{S}^*, 2\mathit{Val}, 2\mathit{Str}, 2U)$ over D together with $\mathbb{S}^*$ and $\times$ consists of a 2-signature category $2\mathbb{S}^*$ over $\mathbb{S}^*$, an indexed category $2\mathit{Val}\colon (2\mathbb{S}^*)^{\mathrm{op}} \to \mathsf{Cat}$ of *2-value variables*, an indexed category $2\mathit{Str}\colon (2\mathbb{S}^*)^{\mathrm{op}} \to \mathsf{Cat}$ of *2-structures*, and an *underlying* indexed functor $2U\colon 2\mathit{Str} \dot\to 2\mathit{Val}$ that are defined as follows:

A *2-signature category* $2\mathbb{S}^*$ over $\mathbb{S}^*$ has as objects *2-signatures* that are pairs $\tau = (\tau_{\mathrm{c}}\colon \Sigma_{\mathrm{b}}(\tau) \to \Sigma_{\mathrm{c}}(\tau), \tau_{\mathrm{cd}}\colon \Sigma_{\mathrm{c}}(\tau) \to \Sigma_{\mathrm{cd}}(\tau))$ of signature morphisms in $\mathbb{S}^*$; and as morphisms *2-signature morphisms* from τ to τ' that are triples $\sigma = (\sigma_{\mathrm{b}}\colon \Sigma_{\mathrm{b}}(\tau) \to \Sigma_{\mathrm{b}}(\tau'), \sigma_{\mathrm{c}}\colon \Sigma_{\mathrm{c}}(\tau) \to \Sigma_{\mathrm{c}}(\tau'), \sigma_{\mathrm{cd}}\colon \Sigma_{\mathrm{cd}}(\tau) \to \Sigma_{\mathrm{cd}}(\tau'))$ of signature morphisms in $\mathbb{S}^*$ such that the following diagram commutes:

$$\begin{tikzcd}
\Sigma_{\mathrm{b}}(\tau) \arrow[r, "\tau_{\mathrm{c}}"] \arrow[d, "\sigma_{\mathrm{b}}"'] & \Sigma_{\mathrm{c}}(\tau) \arrow[r, "\tau_{\mathrm{cd}}"] \arrow[d, "\sigma_{\mathrm{c}}"'] & \Sigma_{\mathrm{cd}}(\tau) \arrow[d, "\sigma_{\mathrm{cd}}"'] \\
\Sigma_{\mathrm{b}}(\tau') \arrow[r, "\tau'_{\mathrm{c}}"] & \Sigma_{\mathrm{c}}(\tau') \arrow[r, "\tau'_{\mathrm{cd}}"] & \Sigma_{\mathrm{cd}}(\tau')
\end{tikzcd}$$

We write $\widehat{\Sigma}(\tau)$ for the push-out object $\Sigma_{\mathrm{cd}}(\tau) +^{\tau_{\mathrm{c}};\tau_{\mathrm{cd}},\tau_{\mathrm{c}};\tau_{\mathrm{cd}}}_{\Sigma_{\mathrm{b}}(\tau)} \Sigma_{\mathrm{cd}}$, and $\overleftarrow{\widehat{\tau}}\colon \Sigma_{\mathrm{cd}}(\tau) \to \widehat{\Sigma}(\tau)$ and $\widehat{\tau}\colon \Sigma_{\mathrm{cd}}(\tau) \to \widehat{\Sigma}(\tau)$ for the first and second push-out morphisms as illustrated in

$$\begin{tikzcd}
 & & \Sigma_{\mathrm{cd}}(\tau) \arrow[dr, "\overleftarrow{\widehat{\tau}}"] & \\
\Sigma_{\mathrm{b}}(\tau) \arrow[r, "\tau_{\mathrm{c}}"] & \Sigma_{\mathrm{c}}(\tau) \arrow[ur, "\tau_{\mathrm{cd}}"] \arrow[dr, "\tau_{\mathrm{cd}}"'] & & \widehat{\Sigma}(\tau) \\
 & & \Sigma_{\mathrm{cd}}(\tau) \arrow[ur, "\widehat{\tau}"'] &
\end{tikzcd}$$

Furthermore, for a 2-signature morphism $\sigma\colon \tau \to \tau'$, we write $\widehat{\sigma}$ for the unique signature morphism from $\widehat{\tau}$ to $\widehat{\tau}'$ that satisfies $\overleftarrow{\widehat{\tau}};\widehat{\sigma} = \sigma_{\mathrm{cd}};\overleftarrow{\widehat{\tau}}'$ and $\widehat{\tau};\widehat{\sigma} = \sigma_{\mathrm{cd}};\widehat{\tau}'$.

Example. The term charter domain $\mathsf{D}^{\leq} = (\mathbb{S}^{\leq}, \mathit{Val}^{\leq}, \mathit{Str}^{\leq}, U^{\leq})$ is 2-able, as described in Sect. 4. For $(\mathbb{S}^{\leq})^*$ we may choose the subcategory of inclusions in $\mathbb{S}^{\leq}$ which has push-outs. The 2-signatures in $2(\mathbb{S}^{\leq})^*$, then, directly correspond to the sorts and declaration extensions; the annotation of attributes with `@pre` is a mere convention.

For a 2-signature $\tau \in |2\mathbb{S}^*|$, the category $2\mathit{Val}(\tau)$ is given by $\mathit{Val}(\widehat{\Sigma}(\tau))$; for a 2-signature morphism $\sigma\colon \tau \to \tau'$, the functor $2\mathit{Val}(\sigma)\colon 2\mathit{Val}(\tau') \to 2\mathit{Val}(\tau)$ by $\mathit{Val}(\widehat{\sigma})$.

For a 2-signature $\tau \in |2\mathbb{S}^*|$, the category $2\mathit{Str}(\tau)$ has as objects pairs of structures $(\overleftarrow{M}, M) \in |\mathit{Str}(\Sigma_{\mathrm{cd}}(\tau))| \times |\mathit{Str}(\Sigma_{\mathrm{cd}}(\tau))|$ with $\mathit{Str}(\tau_{\mathrm{c}};\tau_{\mathrm{cd}})(\overleftarrow{M}) \cong \mathit{Str}(\tau_{\mathrm{c}};\tau_{\mathrm{cd}})(M)$; and as morphisms from $(\overleftarrow{M}_1, M_1)$ to $(\overleftarrow{M}_2, M_2)$ pairs of structure morphisms $(\overleftarrow{\mu}\colon \overleftarrow{M}_1 \to \overleftarrow{M}_2, \mu\colon M_1 \to M_2)$ in $\mathit{Str}(\Sigma_{\mathrm{cd}}(\tau))$ with $\mathit{Str}(\tau_{\mathrm{c}};\tau_{\mathrm{cd}})(\overleftarrow{\mu}) \cong \mathit{Str}(\tau_{\mathrm{c}};\tau_{\mathrm{cd}})(\mu)$. For a 2-signature morphism $\sigma\colon \tau \to \tau'$, the functor $2\mathit{Str}(\sigma)\colon 2\mathit{Str}(\tau') \to 2\mathit{Str}(\tau)$ is defined by $2\mathit{Str}(\sigma)(\overleftarrow{M}, M) = (\mathit{Str}(\sigma_{\mathrm{cd}})(\overleftarrow{M}), \mathit{Str}(\sigma_{\mathrm{cd}})(M))$ and $2\mathit{Str}(\sigma)(\overleftarrow{\mu}, \mu) = (\mathit{Str}(\sigma_{\mathrm{cd}})(\overleftarrow{\mu}), \mathit{Str}(\sigma_{\mathrm{cd}})(\mu))$.

Example. For the previous example, the 2-structures over $2(\mathbb{S}^{\leq})^*$ are given by pairs of Str-structures that share a common basic structure.

Finally, for a 2-signature $\tau \in |2\mathbb{S}^*|$, the underlying functor $2U_\tau\colon 2\mathit{Str}(\tau) \to 2\mathit{Val}(\tau)$ maps a $(\overleftarrow{M}, M) \in |2\mathit{Str}(\tau)|$ to $U_{\widehat{\Sigma}(\tau)}(\overleftarrow{M} \times_{\Sigma_{\mathrm{b}}(\tau)} M)$ and a $(\overleftarrow{\mu}, \mu)\colon (\overleftarrow{M}_1, M_1) \to (\overleftarrow{M}_2, M_2)$ in $2\mathit{Str}(\tau)$ to $U_{\widehat{\Sigma}(\tau)}(\overleftarrow{\mu} \times_{\Sigma_{\mathrm{b}}(\tau)} \mu)$.

By expanding these definitions, we obtain

Proposition. *The 2-term charter* $2\mathsf{D}^* = (2\mathbb{S}^*, 2\mathit{Val}, 2\mathit{Str}, 2U)$ *over the 2-able term charter domain* D *is a term charter domain.*

5.2. 2-Term Charters

Let $\mathsf{D} = (\mathbb{S}, \mathit{Val}, \mathit{Str}, U)$ be a 2-able term charter domain and $\mathfrak{T} = (\mathscr{C}, \nu, \mathit{ext})$ a term charter over D. Let $2\mathsf{D}^* = (2\mathbb{S}^*, 2\mathit{Val}, 2\mathit{Str}, 2U)$ be the 2-term charter domain over D together with a push-out subcategory $\mathbb{S}^*$ of $\mathbb{S}$ and a weak union amalgamation construction $\times$ for Str w.r.t. $\mathbb{S}^*$. The *2-term charter* $(2\mathscr{C}, 2\nu, 2\mathit{ext})$ over $2\mathsf{D}^*$ consists of a lax indexed functor $2\mathscr{C}\colon 2\mathit{Val} \dot\to 2\mathit{Val}$, a lax indexed natural transformation $2\nu\colon 1_{2\mathit{Val}} \dot\to 2\mathscr{C}$, and a family of maps $(2\mathit{ext}_\tau)^{(\overleftarrow{M}, M)}_{\widehat{X}}\colon 2\mathit{Val}(\tau)(\widehat{X}, 2U_\tau(\overleftarrow{M}, M)) \to 2\mathit{Val}(\tau)(2\mathscr{C}_\tau(\widehat{X}), 2U_\tau(\overleftarrow{M}, M))$ for $\tau \in |2\mathbb{S}^*|$, $\widehat{X} \in |2\mathit{Val}(\tau)|$, and $(\overleftarrow{M}, M) \in |2\mathit{Str}(\tau)|$ that are defined as follows:

- For a 2-signature $\tau \in |2\mathbb{S}^*|$, the 2-terms construction and renaming functor $2\mathscr{C}_\tau\colon 2\mathit{Val}(\tau) \to 2\mathit{Val}(\tau)$ is given by $\mathscr{C}_{\widehat{\Sigma}(\tau)}$; for a 2-signature morphism $\sigma\colon \tau \to \tau'$ in $2\mathbb{S}^*$, the 2-terms translating natural transformation $2\mathscr{C}_\sigma\colon 2\mathit{Val}(\sigma); 2\mathscr{C}_\tau \dot\to 2\mathscr{C}_{\tau'}; 2\mathit{Val}(\sigma)$ is given by $\mathscr{C}_{\widehat{\sigma}}$.
- For a 2-signature $\tau \in |2\mathbb{S}^*|$, the 2-embedding natural transformation $2\nu_\tau\colon 1_{2\mathit{Val}(\tau)} \dot\to 2\mathscr{C}_\tau$ is given by $\nu_{\widehat{\Sigma}(\tau)}$.
- For a 2-signature $\tau \in |2\mathbb{S}^*|$, 2-value variables $\widehat{X} \in |2\mathit{Val}(\tau)|$, a 2-structure $(\overleftarrow{M}, M) \in |2\mathit{Str}(\tau)|$, and a 2-valuation $\widehat{\beta}\colon \widehat{X} \to 2U_\tau(\overleftarrow{M}, M)$, the 2-terms evaluation map $(2\mathit{ext}_\tau)^{(\overleftarrow{M}, M)}_{\widehat{X}}(\widehat{\beta})\colon 2\mathscr{C}_\tau(\widehat{X}) \to 2U_\tau(\overleftarrow{M}, M)$ is given by $(\mathit{ext}_{\widehat{\Sigma}(\tau)})^{\overleftarrow{M} \times_{\Sigma_{\mathrm{b}}(\tau)} M}_{\widehat{X}}(\widehat{\beta})$.

Using the requirements on weak union amalgamations we obtain

Proposition. *The 2-term charter* $(2\mathscr{C}, 2\nu, 2\mathit{ext})$ *over the term charter domain* $2\mathsf{D}^*$ *is a term charter.*

Proof. Requirements (V) and (R) for $(2\mathscr{C}, 2\nu, 2\mathit{ext})$ directly follow from the respective requirements for $(\mathscr{C}, \nu, \mathit{ext})$ using push-outs. For (T) we have with the same notation as above

$$\begin{aligned}
& 2\mathscr{C}_\sigma(\widehat{X}'); 2\mathit{Val}(\sigma)((2\mathit{ext}_{\tau'})^{(\overleftarrow{M}', M')}_{\widehat{X}'}(\widehat{\beta}')) \\
={}& \mathscr{C}_{\widehat{\sigma}}(\widehat{X}'); \mathit{Val}(\widehat{\sigma})((\mathit{ext}_{\widehat{\Sigma}(\tau')})^{\overleftarrow{M}' \times_{\Sigma_{\mathrm{b}}(\tau')} M'}_{\widehat{X}'}(\widehat{\beta}')) \\
={}& (\mathit{ext}_{\widehat{\Sigma}(\tau)})^{\mathit{Str}(\widehat{\sigma})(\overleftarrow{M}' \times_{\Sigma_{\mathrm{b}}(\tau')} M')}_{\mathit{Val}(\widehat{\sigma})(\widehat{X}')}(\mathit{Val}(\widehat{\sigma})(\widehat{\beta}')) \\
={}& (\mathit{ext}_{\widehat{\Sigma}(\tau)})^{\mathit{Str}(\sigma_{\mathrm{cd}})(\overleftarrow{M}') \times_{\Sigma_{\mathrm{b}}(\tau)} \mathit{Str}(\sigma_{\mathrm{cd}})(M')}_{\mathit{Val}(\widehat{\sigma})(\widehat{X}')}(\mathit{Val}(\widehat{\sigma})(\widehat{\beta}'))
\end{aligned}$$

$$= (2ext_\tau)^{2Str(\sigma)(\overleftarrow{M}',M')}_{2Val(\sigma)(\widehat{X}')}(2Val(\sigma)(\widehat{\beta}'))\ . \qquad \square$$

EXAMPLE. Consider the 2-term charter domain $2(\mathsf{D}^{\leq})^*$ over the 2-able term charter domain $\mathsf{D}^{\leq}$ as above and the term charter $\mathfrak{T}^{\leq}$ over $\mathsf{D}^{\leq}$. The functor $2\mathscr{T}_\tau^{\leq}$ then constructs the terms over $\widehat{\Sigma}(\tau)$ that include both unadorned and adorned function symbols, the latter for "post-condition time". A valuation $\widehat{\beta}\colon \widehat{X} \to 2U_\tau(\overleftarrow{M}, M)$ indeed is a function $\widehat{\beta}\colon \widehat{X} \to U_{\widehat{\Sigma}(\tau)}(\overleftarrow{M} \times_{\Sigma_b(\tau)} M)$ that can map value variables to values from both the pre-state $\overleftarrow{M}$ and the post-state M.

5.3. Extending 2-Term Charters

We now want to apply the 2-construction to OCL and to include also `allInstances` and `isNew` which, as mentioned above, need special attention. We therefore move to the OCL-closed term charter domain D° and the co-limit $\mathfrak{T}^\mathrm{o} = (\mathfrak{T}^\mathrm{it} \rhd \mathfrak{T}^\mathrm{u})^{\rhd *} = (\mathscr{C}^\mathrm{o}, \nu^\mathrm{o}, ext^\mathrm{o})$ of the iterate and undefinedness term charters without including the all-instances term charter $\mathfrak{T}^\mathrm{a}$. There the basic signature for push-outs and the basic structure for weak union amalgamations is dispensable as always the OCL standard library and its predefined interpretation may be chosen. Still, the construction of 2-term charter domains and 2-term charters is directly applicable, yielding $2(\mathsf{D}^\circ)^*$ and $2\mathfrak{T}^\mathrm{o} = (2\mathscr{C}^\mathrm{o}, 2\nu^\mathrm{o}, 2ext^\mathrm{o})$. For the integration of `allInstances` and `isNew` we can again use the constructions on term charters defined in Sect. 3.3, since $2(\mathsf{D}^\circ)^*$ is a term charter domain and $2\mathfrak{T}^\mathrm{o}$ a term charter. We define a new term charter $(\mathscr{C}^{2\mathrm{o}}, \nu^{2\mathrm{o}}, ext^{2\mathrm{o}})$ over $2(\mathsf{D}^\circ)^*$ that just comprises these two constructs for a $\tau \in |2\mathsf{S}^{\circ *}|$ with $\Sigma_\mathrm{cd}(\tau) = (S_\mathrm{cd}, F_\mathrm{cd})$:

- s`.allInstances()` $\in \mathscr{C}^{2\mathrm{o}}_\tau(\widehat{X})_{\mathtt{Set(}s\mathtt{)}}$ for $s \in |S_\mathrm{cd}|$
- v`.isNew()` $\in \mathscr{C}^{2\mathrm{o}}_\tau(\widehat{X})_{\mathtt{Bool}}$ for $s \in |S_\mathrm{cd}|$ and $v \in \widehat{X}_s$

Their evaluation reads:

- $(ext^{2\mathrm{o}}_\tau)^{(\overleftarrow{M},M)}_{\widehat{X}}(\widehat{\beta})(s\mathtt{.allInstances()}) =$
  $(ext^{\mathrm{a}}_{\Sigma_\mathrm{cd}(\tau)})^{M}_{Val^\circ(\widehat{\tau})(\widehat{X})}(Val^\circ(\widehat{\tau})(\widehat{\beta}))(s\mathtt{.allInstances()})$
- $(ext^{2\mathrm{o}}_\tau)^{(\overleftarrow{M},M)}_{\widehat{X}}(\widehat{\beta})(v\mathtt{.isNew()}) = \begin{cases} tt & \text{if } v \in s^M \setminus s^{\overleftarrow{M}} \\ f\!f & \text{otherwise} \end{cases}$

The rule for s`.allInstances()` only has to take into account the post-state M and we can re-use the evaluation in the term charter $\mathfrak{T}^\mathrm{a}$ for `allInstances` by adapting the signature of the variables $\widehat{X}$ and of the valuation $\widehat{\beta}$ by reducts. The rule for v`.isNew()` only looks at the set difference between the interpretation of the sort s of v in the post-state M and the pre-state $\overleftarrow{M}$.

Now we obtain the full OCL terms for (pre- and) post-conditions by $(2\mathfrak{T}^\mathrm{o} \rhd \mathfrak{T}^{2\mathrm{o}})^{\rhd *}$.

6. Conclusions

Motivated by the OCL and its contract language, we have presented a systematic transition from term charter domains and term charters for evaluating general expressions over a single system state to 2-term charter domains and 2-term charters involving two system states. We have provided conditions that make term charter domains 2-able by means of push-outs and a weak form of amalgamation, and we have also shown that 2-term charter domains and 2-term charters are again term charter domains and term charters, respectively. Term charters provide a modular and compositional framework for expression language design and their extension to 2-term charters thus makes modularisation also available for relational expressions and OCL pre- and post-conditions.

The OCL is a syntactically and semantically heterogeneous language that involves different algebraic and non-algebraic expression constructs as well as different semantic domains for partiality, non-determinism, and non-termination. Although all features can be faithfully captured by term charter domains and term charters as well as put into relation by term charter (domain) morphisms (Knapp & Cengarle 2018), a general procedure for constructively combining different syntactic constructs over different semantic domains is still missing.

Extending OCL contracts that involve two system snapshots, filmstrip models have been introduced by Gogolla et al. (2014) for capturing system dynamics over a series of snapshots. Reasoning about such filmstrips could be enabled by moving from 2-term charters to n-term charters.

The application of the term charter framework to the OCL should be implemented in a tool for making the approach executable and accompanied by an entailment system for verifying OCL invariants and contracts. Moreover, the framework should be integrated into the institution-based "Heterogeneous Tool Set" (Mossakowski et al. 2007) in order to "institutionalise" OCL and to study UML/OCL model consistency (Hilken et al. 2014; Gogolla & Hilken 2015) as put forward by Martin Gogolla.

References

Baumeister, H. (1995). Relations as abstract datatypes: An institution to specify relations between algebras. In P. D. Mosses, M. Nielsen, & M. I. Schwartzbach (Eds.), *Proc. 6th intl. conf. theory and practice of software development (tapsoft 1995)* (Vol. 915, p. 756-771). Springer.

Borceux, F. (1994). *Handbook of categorical algebra. vol. 2: Categories and structures*. Cambridge University Press.

Brucker, A. D., & Wolff, B. (2008). HOL-OCL: A formal proof environment for UML/OCL. In J. L. Fiadeiro & P. Inverardi (Eds.), *Proc. 11th intl. conf. fundamental approaches to software engineering (fase 2008)* (Vol. 4961, p. 97-100). Springer.

Cengarle, M. V., & Knapp, A. (2001). A formal semantics for OCL 1.4. In M. Gogolla & C. Kobryn (Eds.), *Proc. 3rd intl. conf. unified modeling language (uml 2001)* (p. 118-133). Springer.

Cengarle, M. V., & Knapp, A. (2004). OCL 1.4/5 vs. 2.0 expressions — formal semantics and expressiveness. *Softw. Syst. Model.*, *3*(1), 9-30.

Cengarle, M. V., Knapp, A., Tarlecki, A., & Wirsing, M. (2008). A heterogeneous approach to UML semantics. In P. Degano, R. De Nicola, & J. Meseguer (Eds.), *Concurrency, graphs and* *models, essays dedicated to ugo montanari on the occasion of his 65th birthday* (Vol. 5065, p. 383-402). Springer.

Diaconescu, R. (2008). *Institution-independent model theory*. Birkhäuser.

Ehrich, H.-D., Gogolla, M., & Lipeck, U. W. (1989). *Algebraische Spezifikation abstrakter Datentypen*. B. G. Teubner.

Engels, G., & Gogolla, M. (1982). Error handling in algebraic specifications. In H.-D. Ehrich & U. W. Lipeck (Eds.), *Proc. 1st ws. abstract data types*. University of Dortmund.

Gogolla, M., Büttner, F., & Richters, M. (2007). USE: A UML-based specification environment for validating UML and OCL. *Sci. Comput. Program.*, *69*(1–3), 27-34.

Gogolla, M., Hamann, L., Hilken, F., Kuhlmann, M., & France, R. (2014). From application models to filmstrip models: An approach to automatic validation of model dynamics. In H.-G. Fill, D. Karagiannis, & U. Reimer (Eds.), *Modellierung 2014* (p. 273-288). Bonn: Gesellschaft für Informatik e.V.

Gogolla, M., Havakili, H., & Schipke, C. (2020). Advanced features for model visualization in the UML and OCL tool USE. In J. Michael et al. (Eds.), *Comp. proc. short, ws., tools papers modellierung 2020* (Vol. 2542, pp. 203–207). CEUR-WS.org.

Gogolla, M., & Hilken, F. (2015). UML and OCL transformation model analysis: Checking invariant independence. In M. Amrani, E. Syriani, & M. Wimmer (Eds.), *Proc. 4th intl. ws. verification of model transformations* (Vol. 1530, p. 20-27). CEUR-WS.org.

Gogolla, M., & Richters, M. (1997). On constraints and queries in UML. In M. Schader & A. Korthaus (Eds.), *Proc. ws. the unified modeling language — technical aspects and applications* (p. 109-121). Physica-Verlag.

Gogolla, M., & Richters, M. (2002). Development of UML descriptions with USE. In H. Shafazand & A. M. Tjoa (Eds.), *Proc. 1st eurasian conf. information and communication technology (eurasia-ict 2002)* (Vol. 2510, p. 228-238). Springer.

Goguen, J. A., & Burstall, R. M. (1992). Institutions: Abstract model theory for specification and programming. *J. ACM*, *39*, 95-146.

Hilken, F., Niemann, P., Gogolla, M., & Wille, R. (2014). Filmstripping and unrolling: A comparison of verification approaches for UML and OCL behavioral models. In M. Seidl & N. Tillmann (Eds.), *Proc. 8th intl. conf. tests and proofs (tap 2014)* (Vol. 8570, p. 99-116). Springer.

Knapp, A., & Cengarle, M. V. (2015). Institutions for OCL-like expression languages. In R. De Nicola & R. Hennicker (Eds.), *Software, services, and systems — essays dedicated to martin wirsing on the occasion of his retirement from the chair of programming and software engineering* (Vol. 8950, p. 193-214). Springer.

Knapp, A., & Cengarle, M. V. (2018). Term charters. In J. L. Fiadeiro & I. Tutu (Eds.), *Rev. sel. papers 24th ifip wg 1.3 intl. ws. recent trends in algebraic development techniques* (Vol. 11563, p. 119-138). Springer.

Mossakowski, T., Maeder, C., & Lüttich, K. (2007). The heterogeneous tool set. In O. Grumberg & M. Huth (Eds.), *Proc. 13th intl. conf. tools and algorithms for the construction and analysis of systems (tacas 2007)* (Vol. 4424, p. 519-522). Springer.

Object Management Group. (2014). *Object Constraint Language* (Standard No. formal/2014-02-03). OMG. Retrieved from http://www.omg.org/spec/OCL/2.4

Pawłowski, W. (1996). Context institutions. In *Sel. papers 11th ws. specification of abstract data types & 8th compass ws. recent trends in data type specifications* (Vol. 1130, p. 436-457). Springer.

Reggio, G., Astesiano, E., & Choppy, C. (2003). CASL-LTL*: A* CASL *extension for dynamic reactive systems. Version 1.0. Summary* (Technical Report No. DISI-TR-03-36). Univ. Genova.

Richters, M., & Gogolla, M. (1998). On formalizing the UML object constraint language OCL. In T. W. Ling, S. Ram, & M. Lee (Eds.), *Proc. 17th intl. conf. conceptual modeling (er 1998)* (Vol. 1507, p. 449-464). Springer.

Sannella, D., & Tarlecki, A. (2012). *Foundations of algebraic specification and formal software development*. Springer.

Tarlecki, A., Burstall, R. M., & Goguen, J. A. (1991). Some fundamental algebraic tools for the semantics of computation, part 3: Indexed categories. *Theo. Comp. Sci.*, *91*, 239-264.

Wolter, U., & Martini, A. (1997). Shedding new light in the world of logical systems. In *Proc. 7th intl. conf. category theory and computer science (ctcs 1997)* (Vol. 1290, p. 159-176). Springer.

About the authors

Alexander Knapp is professor for the foundations of software and systems engineering at the University of Augsburg. His research interests include formal methods, modal and hybrid logics, the semantics of modelling and programming languages, and soft constraints. You can contact the author at knapp@isse.de or visit https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/swtsse/.

María Victoria Cengarle has been a researcher at the Ludwig-Maximilians-University Munich, the Fraunhofer Institute for Experimental Software Engineering, the Technical University of Munich, and the fortiss research institute. Her interests include algebraic specifications, formal semantics of programming and modelling languages, and compositional verification methods. You can contact the author at mv.cengarle@gmail.com.