Institutions for OCL-like expression languages

Alexander Knapp, María Victoria Cengarle

Angaben zur Veröffentlichung / Publication details:

Knapp, Alexander, and María Victoria Cengarle. 2015. "Institutions for OCL-like expression languages." *Lecture Notes in Computer Science* 8950: 193–214.
https://doi.org/10.1007/978-3-319-15545-6_14.

Institutions for OCL-Like Expression Languages

Alexander Knapp[1] and María Victoria Cengarle[2]

[1] Universität Augsburg, Germany
knapp@informatik.uni-augsburg.de
[2] Technische Universität München, Germany
cengarle@in.tum.de

Abstract. In 2008, Martin Wirsing initiated the project of conceiving the "Unified Modeling Language" (UML) as a heterogeneous modelling language. He proposed to use the theory of heterogeneous institutions for providing individual semantics to each sub-language, that can then be integrated using institution (co-)morphisms. In particular, the proposal allows for seamlessly capturing the notorious semantic variation points of UML with mathematical rigour. In this line of research, we contribute an institutional framework for the "Object Constraint Language" (OCL), UML's language for expressing constraints.

1 Introduction

The "Unified Modeling Language" (UML), in its inception and according to its own definition, "is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system" [1, p. XV]. The UML, on the one hand, has been repeatedly criticized because of its lack of formal semantics. On the other hand, UML has been praised for being the "lingua franca" that acts as an Esperanto among stakeholders, be these application domain experts, system designers, program developers, or clients.[1] The scientific community has spent some effort in providing UML with a formal semantics that, among other things, allows for the rigorous verification of properties of interest of the software system under consideration. These efforts, however, have not been crowned with the success they might deserve, probably because they impose a "straitjacket" to UML users, what in its turn is against a stance advocated by the UML language designers that UML be somehow free in the way it should be understood. Indeed, the standard foresees so-called semantic variation points that allow language users to interpret language constructs differently.

Martin Wirsing, therefore, proposed a heterogeneous approach that allows UML users the definition of the preferred semantics to the individual UML sub-languages, and is such that the composition of those languages and their attached semantics permits compositional proofs; see [4] and also [2]. The proposed approach builds on the abstract model theory framework of institutions [8], where each sub-language is captured as an institution. Originally, institutions have been devised for formalizing logical systems with their signatures, sentences, structures, and satisfaction relation, imposing only minimal constraints, namely that satisfaction be invariable under change of syntax. Formally,

[1] Empirical evidence for the various, but rather limited usages of the UML in industrial practice has been gathered by Petre [18].

an *institution* $(\mathrm{Sig}, Str, Sen, \models)$ is given by (i) a category Sig whose objects are called *signatures*; (ii) a contravariant functor $Str : \mathrm{Sig}^{\mathrm{op}} \to \mathrm{Cat}$, called the *structure functor*, from Sig to Cat, the category of categories; (iii) a functor $Sen : \mathrm{Sig} \to \mathrm{Set}$, called the *sentence functor*, from Sig to Set, the category of sets; and (iv) a family $\models = \{\models_\Sigma\}_{\Sigma \in \mathrm{Sig}}$ of *satisfaction relations* between Σ-structures $M \in Str(\Sigma)$ and Σ-sentences $\varphi \in Sen(\Sigma)$, such that for each $\sigma : \Sigma \to \Sigma'$ in Sig, $M' \in Str(\Sigma)$, and $\varphi \in Sen(\Sigma)$, the following *satisfaction condition* holds:

$$Str(\sigma)(M') \models_\Sigma \varphi \iff M' \models_{\Sigma'} Sen(\sigma)(\varphi) \ .$$

For the application to UML sub-languages, the syntactic elements available in each sub-language are rendered as signatures, their meaning as structures, and their possible combinations as sentences. Semantic variation points or particular domain-specific usages of a sub-language lead to different institutions. The framework of institutions provides a rich family of institution (co-)morphisms for relating institutions in terms of embeddings and projections. For UML sub-languages expressed as institutions, these (co-)morphisms can be applied to express refinements and consistency conditions between sub-languages and different resolutions of semantic variation points.

The aim of this work is to give a definition of the "Object Constraint Language" (OCL [16]) that satisfies the conditions associated with institutions. The OCL provides a textual expression language for navigating through UML models, specifying guards and pre-/post-conditions, and for defining constraints, like invariants, on model elements. In the UML specification [15] the OCL is used for specifying well-formedness rules on models. Though strictly speaking not a UML sub-language, the OCL constitutes a natural modelling ingredient complementing the visual notation of the UML.

The first difficulty, that at first sight seems an incompatibility, is that OCL focuses on terms and not on truth. This way, for instance, a three-valued logic is possible. So, the core property of institutions, namely the satisfaction condition, needs be defined for terms, in the form of an evaluation condition. In fact, this is already the case for, e.g., classical first-order predicate logic with function symbols (for term construction), predicate symbols (for atom construction), and logical connectives and quantifiers (for formula construction). This means, the property called for is satisfied in the classical setting and needs only be mimicked for a definition of OCL terms. Thus, it should be possible to use some formal OCL expression semantics, like, e.g., [3], and derive an institution directly. However, it turns out that some OCL constructs like `if-then-else`, `iterate`, or `allInstances` are more naturally handled as special term formers than as function symbols directly. This motivates a two-level language definition for OCL terms, namely the already mentioned function symbols and a construction functor for the term formers. Using the language of indexed categories (see Sect. 2), we define the notion of term charters for capturing such general term languages, their evaluation, and, in particular, an evaluation condition in Sect. 3. We also show how languages defined by means of term charters can be turned into an institution.[2]

A further characteristics of OCL is that it is constituted by many sub-theories: order-sortedness, non-strict evaluation, three-valued logic, non-determinism, etc. For particular

[2] The manuscript accompanying this article, that shows the proof of every assertion here, can be found in [11].

domains, different combinations or extensions of the sub-theories may be useful, see, e.g., [3,12]. For this reason, and in order to provide the modelling language designer with a powerful tool, means are defined that allow for a compositional definition of a term-based constraint language. Each sub-language can be defined separately, and it is possible to build different constraint languages, that contain the needed theories for the situation at hand, by putting up different sub-theories. Therefore, a further goal of this work is the elucidation of a (meta-)theory for the compositional integration of those sub-theories. This is akin to the specification-building operators defined by Martin Wirsing in [21], only on a meta-level. Examples of OCL theories are shown in Sect. 4, means for their composition are presented in Sect. 5.

2 Indexed Categories

We briefly recall the basic notions of indexed categories (see, e.g., [20]) mainly for fixing the notation.

An *indexed category* N over an index category I is a functor $N : I^{\mathrm{op}} \to \mathrm{Cat}$. Given an I-indexed category $N : I^{\mathrm{op}} \to \mathrm{Cat}$, the *Grothendieck category* $\mathcal{G}(N)$ over N has as objects the pairs $\langle i, O\rangle$ with $i \in |I|$ and $O \in |N(i)|$, and as morphisms from $\langle i, O\rangle$ to $\langle i', O'\rangle$ the pairs $\langle u, o\rangle$ with $u \in I(i, i')$ and $o \in N(i)(O, N(u)(O'))$; the identity morphism on $\langle i, O\rangle$ is $\langle 1_i, 1_O\rangle$, the composition of morphisms $\langle u, o\rangle : \langle i, O\rangle \to \langle i', O'\rangle$ and $\langle u', o'\rangle : \langle i', O'\rangle \to \langle i'', O''\rangle$ is $\langle u, o\rangle; \langle u', o'\rangle = \langle u; u', o; N(u)(o')\rangle$.

The *projection functor* π_N from $\mathcal{G}(N)$ to I is defined by $\pi_N(\langle i, O\rangle) = i$ and $\pi_N(\langle u, o\rangle) = u$. For an $i \in |I|$, $\mathcal{G}(N)(i)$ denotes the sub-category of $\mathcal{G}(N)$ with objects $\langle i, O\rangle$ and morphisms $\langle 1_i, o\rangle$.

A morphism $u : i \to i'$ in I induces the *reduct functor* $-|_N u : \mathcal{G}(N)(i') \to \mathcal{G}(N)(i)$ with $\langle i', O'\rangle|_N u = \langle i, N(u)(O')\rangle$ and $\langle 1_{i'}, o'\rangle|_N u = \langle 1_i, N(u)(o')\rangle$. For $\langle i', O'\rangle \in |\mathcal{G}(N)|$, $u : i \to i'$ also induces the *forward morphism* $u^{|_N\langle i', O'\rangle} = \langle u, 1_{N(u)(O')}\rangle : \langle i, N(u)(O')\rangle \to \langle i', O'\rangle$; in particular, $u^{|_N -} : -|_N u \dot{\to} 1_{\mathcal{G}(N)(i')}$ is a natural transformation. Each morphism $\langle u, o\rangle : \langle i, O\rangle \to \langle i', O'\rangle$ can be uniquely factorized as $\langle u, o\rangle = \langle 1_i, o\rangle; u^{|_N\langle i', O'\rangle}$ with $\langle 1_i, o\rangle : \langle i, O\rangle \to \langle i', O'\rangle|_N u$; we denote $\langle 1_i, o\rangle$ by $\langle u, o\rangle|_N$.

An *indexed functor* F from an I-indexed category M to an I-indexed category N is a natural transformation $F : M \dot{\to} N$. The *Grothendieck functor* $\mathcal{G}(F) : \mathcal{G}(M) \to \mathcal{G}(N)$ over F is defined by $\mathcal{G}(F)(\langle i, O\rangle) = \langle i, F_i(O)\rangle$ and $\mathcal{G}(F)(\langle u, o\rangle : \langle i, O\rangle \to \langle i', O'\rangle) = \langle u, F_i(o)\rangle : \langle i, F_i(O)\rangle \to \langle i', F_{i'}(O')\rangle$.

Lemma 1. *Let $M, N : I^{\mathrm{op}} \to \mathrm{Cat}$ be indexed categories and $F : M \dot{\to} N$ an indexed functor. Let $u : i \to i'$ in I and $\langle i', O'\rangle \in |\mathcal{G}(M)|$. Then*

(1) $\mathcal{G}(F); (-|_N u) = (-|_M u); \mathcal{G}(F)$;
(2) $u^{|_N \mathcal{G}(F)(\langle i', O'\rangle)} = \mathcal{G}(F)(u^{|_M \langle i', O'\rangle})$;
(3) $\pi_M = \mathcal{G}(F); \pi_N$.

3 Term Charters

The core part of the OCL is an expression or term language, where formulae are captured as Boolean expressions that can then be used as guards, invariants, or pre-/post-conditions. When institutionalizing OCL we thus want to focus on its expressions in

their own right and extend the satisfaction condition for formulae to an "evaluation condition" for terms. We therefore employ a framework that mimics and generalizes classical term evaluation with valuations for variables [21]: Terms over a signature are built by a construction functor $\mathscr{C}$ that takes values as variables X from a signature-indexed category Val and yields the term language, again in Val. Evaluation of a term over a given valuation β is described by a lifting $(\beta)^\natural_M$ from $\mathscr{C}(X)$ to the values in a structure M from a signature-indexed category Str. The evaluation condition requires that evaluation is invariant w.r.t. signature changes.

We call our evaluation framework "term charters", as it is inspired by the notion of charters [7] for constructing institutions. A charter is given by an adjunction $(U, F, \eta, (-)^\sharp)$ between a category of signatures Sign and a category of syntactic systems Syn, a ground object $G \in |\mathrm{Syn}|$ and a base functor $B : \mathrm{Syn} \to \mathrm{Set}$ with $B(G) = \{ff, tt\}$. An institution is obtained from a charter by using Sign as the signatures, and defining, for each $\Sigma \in |\mathrm{Sign}|$, the Σ-structures as the Sign-morphisms $m : \Sigma \to U(G)$, the Σ-sentences as $B(F(\Sigma))$, and the satisfaction relation by $m \models_\Sigma e$ if, and only if $B(m^\sharp)(e) = tt$.[3] Term charters mainly deviate from charters in making the variables of terms explicit in the indexed category Val such that evaluation by means of $(-)^\sharp$ is shifted to taking into account valuations. In charters, these valuations are contained in the single semantic ground object that also comprises all possible interpretations of the signatures, necessitating a "Procrustean ground signature" [7, p. 324] of this ground object which sometimes may not seem the most natural choice; in term charters the ground object is split into several semantic structure objects from the indexed category Str representing different interpretations. Finally, term charters do not insist on an adjunction between the syntactic domain $\mathcal{G}(Val)$ and the semantic domain $\mathcal{G}(Str)$ which makes them applicable in situations where the evaluation structures should only consist in standard interpretations but the syntactic domain may lead to non-standard interpretations, as, e.g., for pre-defined data types or higher-order functions. However, we show below that such an adjunction indeed induces a term charter.

3.1 Term Charter Domains and Term Charters

A term charter is defined over a term charter domain that fixes the signatures, the values and variables, the semantic structures, and how the values are extracted from a structure. A term charter then adds how terms or expressions over variables are constructed and how they are evaluated over the values of a structure. We first give the formal definition and then illustrate the notion of term charters by means of order-sorted algebras [21].

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

A *term charter* $\mathfrak{T} = (\mathscr{C}, \nu, (-)^\natural)$ over a term charter domain $(\mathbb{S}, Val, Str, U)$ is given by a *construction* functor $\mathscr{C} : \mathcal{G}(Val) \to \mathcal{G}(Val)$ with $\pi_{Val} = \mathscr{C}; \pi_{Val}$; an *embedding*

[3] In fact, an adjunction for a charter can be obtained systematically when using the notion of parchments [7,17] that induce a suitable category of syntactic systems as the Grothendieck category $\mathcal{G}(Syn)$ with $Syn(\Sigma) = Alg(Lang(\Sigma))$ where $Lang$ is a functor from the signatures to (many-sorted) algebraic signatures and the functor Alg yields the (many-sorted) algebras over an algebraic signature.

natural transformation $\nu : 1_{\mathcal{G}(Val)} \dot{\to} \mathscr{C}$ with $\nu_X : X \to \mathscr{C}(X)$ in $\mathcal{G}(Val)(\pi_{Val}(X))$; and a $|\mathcal{G}(Str)|$-family $(-)^\natural = ((-)^\natural{}_M)_{M \in |\mathcal{G}(Str)|}$ associating for each $\Sigma \in |\mathbb{S}|$ and $M \in |\mathcal{G}(Str)(\Sigma)|$ to each morphism $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$ a morphism $(\beta)^\natural{}_M : \mathscr{C}(X) \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$ such that

– for all $\Sigma \in \mathbb{S}$, $M \in |\mathcal{G}(Str)(\Sigma)|$, $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$, and $\xi : Y \to X$ in $\mathcal{G}(Val)(\Sigma)$ the following diagrams commute:

(C)
$$\begin{array}{ccc} X & \xrightarrow{\nu_X} & \mathscr{C}(X) \\ & {\scriptstyle\beta}\searrow & \downarrow{\scriptstyle(\beta)^\natural{}_M} \\ & & \mathcal{G}(U)(M) \end{array}$$

(K)
$$\begin{array}{ccc} \mathscr{C}(Y) & \xrightarrow{\mathscr{C}(\xi)} & \mathscr{C}(X) \\ & {\scriptstyle(\xi;\beta)^\natural{}_M}\searrow & \downarrow{\scriptstyle(\beta)^\natural{}_M} \\ & & \mathcal{G}(U)(M) \end{array}$$

– for all $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$, $M' \in |\mathcal{G}(Str)(\Sigma')|$, and $\beta' : X' \to \mathcal{G}(U)(M')$ in $\mathcal{G}(Val)(\Sigma')$ the following diagram commutes:

(E)
$$\begin{array}{ccc} \mathscr{C}(X'|_{Val}\sigma) & \xrightarrow{\mathscr{C}(\sigma^{|_{Val}X'})} & \mathscr{C}(X') \\ \downarrow{\scriptstyle(\beta'|_{Val}\sigma)^\natural{}_{M'|_{Str}\sigma}} & & \downarrow{\scriptstyle(\beta')^\natural{}_{M'}} \\ \mathcal{G}(U)(M'|_{Str}\sigma) & \xrightarrow[\mathcal{G}(U)(\sigma^{|_{Str}M'})]{} & \mathcal{G}(U)(M') \end{array}$$

Requirement (E) is called the *evaluation condition* expressing that evaluation is invariant w.r.t. signature changes. Condition (C) and (K) ensure that valuations are respected by evaluation and that evaluation is compatible with variable renaming.

Example 1. In order to illustrate term charters, we reformulate order-sorted algebras and their terms. For establishing a suitable term charter domain, we first have to fix order-sorted signatures, value domains, and structures.

The category $\mathbb{S}^\leq$ of *order-sorted signatures* has as objects the pairs (S, D) with $S = (|S|, \leq_S)$ a partial order for the *sorts* and $D = (|D|, \delta_D)$ *function declarations* with $\delta_D : |S|^* \times |S| \to \mathcal{P}(|D|)$; and as morphisms pairs $(\gamma, \rho) : (S, D) \to (S', D')$ of a monotone function on the sorts and a sort-compatible function renaming.

For a $\Sigma = (S, D)$, a *Σ-value domain* V consists of a family $(V_s)_{s \in |S|}$ of *values* respecting sub-sorting, i.e., $V_s \subseteq V_{s'}$ if $s \leq_S s'$; and a *Σ-value domain morphism* $\omega : V \to V'$ is given by a family of mappings $\omega = (\omega_s : V_s \to V'_s)_{s \in |S|}$. Similarly, a *$\Sigma$-structure* (V, E) consists of a Σ-value domain V and a family E of *evaluation* functions $E = (E_{\overline{s},s})_{\overline{s} \in |S|^*, s \in |S|}$, where $E_{\overline{s},s} : \delta_D(\overline{s}, s) \to (V_{\overline{s}} \to V_s)$, that is, E assigns to each function type in $|D|$ a set of functions on the corresponding values; and a *Σ-structure morphism* $\omega : (V, E) \to (V', E')$ is given by a Σ-value domain morphism $\omega : V \to V'$ satisfying the *homomorphism condition* $\omega_s(E_{\overline{s},s}(d)(\vec{v})) = E'_{\overline{s},s}(d)(\omega_{\overline{s}}(\vec{v}))$. The indexed categories $Val^\leq, Str^\leq : (\mathbb{S}^\leq)^{\mathrm{op}} \to \mathrm{Cat}$ map each Σ to $Val^\leq(\Sigma)$ and $Str^\leq(\Sigma)$, respectively, and each order-sorted signature morphism to the usual renaming reduct functors. The indexed functor $U^\leq : Str^\leq \dot{\to} Val^\leq$ "forgets" the evaluation functions of a structure.

We thus obtain the term charter domain $(\mathbb{S}^\leq, Val^\leq, Str^\leq, U^\leq)$. For a term charter for order-sorted terms, we now address term construction and evaluation.

The construction functor $\mathscr{C}^{\leq} : \mathcal{G}(Val^{\leq}) \to \mathcal{G}(Val^{\leq})$ assigns to $\langle \Sigma, X \rangle \in |\mathcal{G}(Val^{\leq})|$ with $\Sigma = (S, D)$ the value domain $\mathscr{C}^{\leq}(\langle \Sigma, X \rangle) = \langle \Sigma, V_X^{\leq} \rangle$ such that for each $s \in |S|$ the values in $V_{X,s}^{\leq}$ are given inductively by

– $x \in V_{X,s}^{\leq}$ for $x \in X_s$;
– $d(\vec{v}) \in V_{X,s'}^{\leq}$ for all $s' \geq_S s$ if $d \in \delta_D(\overline{s}, s)$ and $\vec{v} \in V_{X,\overline{s}}^{\leq}$.

For the morphisms in $\mathcal{G}(Val^{\leq})$, $\mathscr{C}^{\leq}$ yields the corresponding renaming morphism in $\mathcal{G}(Val^{\leq})$. As natural transformation $\nu^{\leq} : 1_{\mathcal{G}(Val^{\leq})} \dot{\to} \mathscr{C}^{\leq}$ for *embedding* values or variables into the order-sorted terms we may simply choose the inclusions.

For *evaluating* order-sorted terms over a structure $M = \langle \Sigma, (V, E) \rangle$ in $|\mathcal{G}(Str^{\leq})|$ given a *valuation* $\beta = \langle 1_\Sigma, \beta^{\leq} \rangle : \langle \Sigma, X \rangle \to \mathcal{G}(U^{\leq})(\langle \Sigma, (V, E) \rangle)$ define $(\beta)^{\natural_M^{\leq}} = \langle 1_\Sigma, (\beta^{\leq})^{\natural_M^{\leq}} \rangle : \mathscr{C}^{\leq}(\langle \Sigma, X \rangle) \to \langle \Sigma, (V, E) \rangle$ inductively by

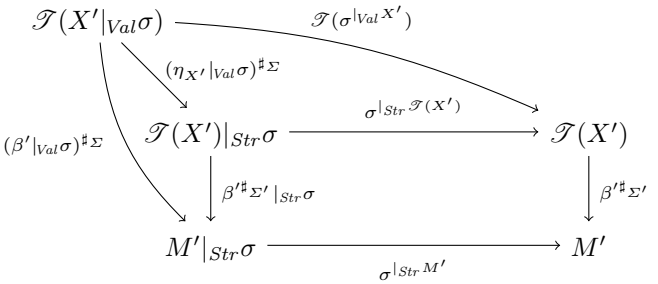
– $(\beta^{\leq})_s^{\natural_M^{\leq}}(x) = \beta_s^{\leq}(x)$ for $x \in X_s$;
– $(\beta^{\leq})_s^{\natural_M^{\leq}}(d(\vec{v})) = E_{\overline{s},s}(d)((\beta^{\leq})_{\overline{s}}^{\natural_M^{\leq}})(\vec{v}))$.

With these definitions, the term charter conditions (C), (K) and, (E) can be checked straightforwardly by induction. Thus we obtain the *order-sorted term charter* $(\mathscr{C}^{\leq}, \nu^{\leq}, (-)^{\natural^{\leq}})$ over the term charter domain $(\mathbb{S}^{\leq}, Val^{\leq}, Str^{\leq}, U^{\leq})$. □

3.2 Term Charters from Adjunctions

The concrete construction of a term charter often involves quite many routine checks, as already illustrated by the previous example of the order-sorted term charter. In the special situation of an adjunction between the syntactic side of Val and the semantic side of Str this effort can be avoided completely.

In fact, let $\mathsf{D} = (\mathbb{S}, Val, Str, U)$ be a term charter domain and assume that $(\mathcal{G}(U), \mathscr{T}, \eta, (-)^\sharp)$ forms an adjunction (expressed as a free construction [19]) with the functor $\mathscr{T} : \mathcal{G}(Val) \to \mathcal{G}(Str)$ satisfying $\pi_{Val} = \mathscr{T}; \pi_{Str}$ the left-adjoint to $\mathcal{G}(U)$, the natural transformation $\eta : 1_{\mathcal{G}(Val)} \dot{\to} \mathscr{T}; \mathcal{G}(U)$ with $\eta_X : X \to \mathcal{G}(U)(\mathscr{T}(X))$ in $\mathcal{G}(Val)(\pi_{Val}(X))$ the unit, and the $\mathbb{S}$-family $(-)^\sharp = ((-)^{\sharp_\Sigma})_{\Sigma \in |\mathbb{S}|}$ associating for each $\sigma \in |\mathbb{S}|$ and $M \in |\mathcal{G}(Str)(\Sigma)|$ to each morphism $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$ a morphism $\beta^{\sharp_\Sigma} : \mathscr{T}(X) \to M$ in $\mathcal{G}(Str)(\Sigma)$ the lifting. Then it can be shown that for each $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$, $M' \in |\mathcal{G}(Str)(\Sigma')|$, and $\beta' : X' \to \mathcal{G}(U)(M')$ in $\mathcal{G}(Val)(\Sigma')$ the following diagram expressing the evaluation condition commutes:



Using this form of the evaluation condition we obtain

Proposition 1. *Let $(\mathcal{G}(U), \mathscr{T}, \eta, (-)^\sharp)$ form an adjunction. Then $(\mathscr{T}; \mathcal{G}(U), \eta, (-)^\natural)$ with $(\beta)^\natural{}_M = \mathcal{G}(U)(\beta^{\sharp_\Sigma})$ for each $\Sigma \in |\mathbb{S}|$, $X \in |\mathcal{G}(Val)(\Sigma)|$, $M \in |\mathcal{G}(Str)(\Sigma)|$, and $\beta : X \to \mathcal{G}(U)(M)$ is a term charter.* □

3.3 Constructing an Institution from a Term Charter

Let $\mathfrak{T} = (\mathscr{C}, \eta, (-)^\natural)$ be a term charter over the term charter domain $(\mathbb{S}, Val, Str, U)$. Let $\mathcal{U}_{Val} : \mathcal{G}(Val) \to \mathrm{Set}$ be a functor such that $\mathcal{U}_{Val}(\sigma^{|_{Val}X'})$ is the inclusion map from $\mathcal{U}_{Val}(X'|_{Val}\sigma)$ to $\mathcal{U}_{Val}(X')$ for $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}$ and $X' \in |\mathcal{G}(Val)(\Sigma')|$, and the *semantic truth value* $* \in \mathcal{U}_{Val}(X)$ for all $X \in |\mathcal{G}(Str); \mathcal{G}(U)| \subseteq |\mathcal{G}(Val)|$.

- Define the category $\mathrm{Sig}^{\mathcal{U}_{Val}}_{\mathfrak{T}}$ as $\mathcal{G}(Val)$.
- Define the functor $Sen^{\mathcal{U}_{Val}}_{\mathfrak{T}} : \mathrm{Sig}^{\mathcal{U}_{Val}}_{\mathfrak{T}} \to \mathrm{Set}$ as $\mathscr{C}; \mathcal{U}_{Val}$.
- Define the functor $Str^{\mathcal{U}_{Val}}_{\mathfrak{T}} : \mathcal{G}(Val)^{\mathrm{op}} \to \mathrm{Cat}$ as
  - the category $Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X)$, where $\Sigma = \pi_{Val}(X)$, with the class of objects the pairs (M, β) with $M \in |\mathcal{G}(Str)(\Sigma)|$ and $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$, and the morphisms $\mu : (M_1, \beta_1) \to (M_2, \beta_2)$ where $\mu \in \mathcal{G}(Str)(\Sigma)(M_1, M_2)$ and $\beta_i : X \to \mathcal{G}(U)(M_i)$ for $1 \leq i \leq 2$ such that $\beta_1; \mathcal{G}(U)(\mu) = \beta_2$;
  - the functor $Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(\xi : X \to X') : Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X') \to Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X)$, where $\sigma = \pi_{Val}(\xi)$, with

$$Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(\xi)(M', \beta') = (M'|_{Str}\sigma, \xi|_{Val}; \beta'|_{Val}\sigma)$$
$$Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(\xi)(\mu' : (M'_1, \beta'_1) \to (M'_2, \beta'_2)) = \mu'|_{Str}\sigma\ .$$

    This is well-defined, since $\beta'_1; \mathcal{G}(U)(\mu') = \beta'_2$ and hence also $\xi|_{Val}; \beta'_1|_{Val}\sigma$; $\mathcal{G}(U)(\mu')|_{Val}\sigma = \xi|_{Val}; \beta'_1|_{Val}\sigma; \mathcal{G}(U)(\mu'|_{Str}\sigma) = \xi|_{Val}; \beta'_2|_{Val}\sigma$.
- Define the family of relations $(\models^{\mathcal{U}_{Val}}_{\mathfrak{T},X})_{X \in |\mathrm{Sig}^{\mathcal{U}_{Val}}_{\mathfrak{T}}|}$ with $\models^{\mathcal{U}_{Val}}_{\mathfrak{T},X} \subseteq |Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X)| \times |Sen^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X)|$ by

$$(M, \beta) \models^{\mathcal{U}_{Val}}_{\mathfrak{T},X} \varphi \quad \text{iff} \quad \mathcal{U}_{Val}((\beta)^\natural{}_M)(\varphi) = *\ .$$

Proposition 2. $(\mathrm{Sig}^{\mathcal{U}_{Val}}_{\mathfrak{T}}, Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}, Sen^{\mathcal{U}_{Val}}_{\mathfrak{T}}, \models^{\mathcal{U}_{Val}}_{\mathfrak{T}})$ *is an institution.*

Proof. We have to show the satisfaction condition

$$Str^{\mathcal{U}_{Val}}_{\mathfrak{T}}(\xi)(M', \beta') \models^{\mathcal{U}_{Val}}_{\mathfrak{T},X} \varphi \quad \text{iff} \quad (M', \beta') \models^{\mathcal{U}_{Val}}_{\mathfrak{T},X'} Sen^{\mathcal{U}_{Val}}_{\mathfrak{T}}(\xi)(\varphi)$$

with $\varphi \in Sen^{\mathcal{U}_{Val}}_{\mathfrak{T}}(X)$, $\xi : X \to X'$ and $\beta' : X' \to \mathcal{G}(U)(M')$. It suffices to prove

$$\mathcal{U}_{Val}((\xi|_{Val}; \beta'|_{Val}\sigma)^{\natural_{M'|_{Str}\sigma}})(\varphi) = \mathcal{U}_{Val}(\mathscr{C}(\xi; (\beta')^{\natural_{M'}}))(\varphi)$$

with $\sigma = \pi_{Val}(\xi)$. We have

$$(\xi|_{Val}; \beta'|_{Val}\sigma)^{\natural_{M'|_{Str}\sigma}}; \sigma^{|_{Val}\mathcal{G}(U)(M')} \overset{(K)}{=}$$
$$\mathscr{C}(\xi|_{Val}); (\beta'|_{Val}\sigma)^{\natural_{M'|_{Str}\sigma}}; \sigma^{|_{Val}\mathcal{G}(U)(M')} \overset{(E)}{=} \mathscr{C}(\xi|_{Val}); \mathscr{C}(\sigma^{|_{Val}X'}); (\beta')^{\natural_{M'}} =$$
$$\mathscr{C}(\xi|_{Val}; \sigma^{|_{Val}X'}); (\beta')^{\natural_{M'}} = \mathscr{C}(\xi); (\beta')^{\natural_{M'}} .$$

The image $\mathcal{U}_{Val}(\sigma^{|_{Val}\mathcal{G}(U)(M')})$ of the forward morphism is an inclusion map. Therefore,

$$\mathcal{U}_{Val}((\xi|_{Val}; \beta'|_{Val}\sigma)^{\natural_{M'|_{Str}\sigma}}; \sigma^{|_{Val}\mathcal{G}(U)(M')})(\varphi) =$$
$$\mathcal{U}_{Val}((\xi|_{Val}; \beta'|_{Val}\sigma)^{\natural_{M'|_{Str}\sigma}})(\varphi) .$$

□

Now additionally assume that for each $\Sigma \in |\mathbb{S}|$ there is an object X^Σ that is initial in $\mathcal{G}(Val)(\Sigma)$. Then, for each each $X \in |\mathcal{G}(Val)|$, there is a unique morphism $\xi^X : X^\Sigma \to X$ in $\mathcal{G}(Val)(\Sigma)$. In particular, for each $M \in |\mathcal{G}(Str)(\Sigma)|$, there is a unique morphism $\beta^\Sigma : X^\Sigma \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$. In this case, we can define a more "classical" institution from the term charter $\mathfrak{T} = (\mathscr{C}, \nu, (-)^\natural)$ as follows:

– Define the category $\mathrm{CSig}_{\mathfrak{T}}^{\mathcal{U}_{Val}}$ as $\mathbb{S}$.
– Define the functor $CSen_{\mathfrak{T}}^{\mathcal{U}_{Val}} : \mathrm{CSig}_{\mathfrak{T}}^{\mathcal{U}_{Val}} \to \mathrm{Set}$ as

$$CSen_{\mathfrak{T}}^{\mathcal{U}_{Val}}(\Sigma) = \mathcal{U}_{Val}(\mathscr{C}(X^\Sigma)) \quad \text{and}$$
$$CSen_{\mathfrak{T}}^{\mathcal{U}_{Val}}(\sigma : \Sigma \to \Sigma') = \mathcal{U}_{Val}(\mathscr{C}(\xi^{X^{\Sigma'}|_{Val}\sigma}; \sigma^{|_{Val}X^{\Sigma'}})) .$$

– Define the functor $CStr_{\mathfrak{T}}^{\mathcal{U}_{Val}} : (\mathrm{CSig}_{\mathfrak{T}}^{\mathcal{U}_{Val}})^{\mathrm{op}} \to \mathrm{Cat}$ as $Str : \mathbb{S}^{\mathrm{op}} \to \mathrm{Cat}$.
– Define the family of relations $(\models_{\mathfrak{T},\Sigma}^{\mathcal{U}_{Val}})_{\Sigma \in |\mathrm{CSig}_{\mathfrak{T}}^{\mathcal{U}_{Val}}|}$ with $\models_{\mathfrak{T},\Sigma}^{\mathcal{U}_{Val}} \subseteq |CStr_{\mathfrak{T}}^{\mathcal{U}_{Val}}(\Sigma)| \times |CSen_{\mathfrak{T}}^{\mathcal{U}_{Val}}(\Sigma)|$ by

$$M \models_{\mathfrak{T},\Sigma}^{\mathcal{U}_{Val}} \varphi \quad \text{iff} \quad \mathcal{U}_{Val}((\beta^\Sigma)^{\natural_M})(\varphi) = * .$$

Corollary 1. $(\mathrm{CSig}_{\mathfrak{T}}^{\mathcal{U}_{Val}}, CStr_{\mathfrak{T}}^{\mathcal{U}_{Val}}, CSen_{\mathfrak{T}}^{\mathcal{U}_{Val}}, \models_{\mathfrak{T}}^{\mathcal{U}_{Val}})$ *is an institution.* □

4 OCL Terms and Evaluation

The main use of OCL for UML models is navigation through a system's maze of objects and links. A domain for this task is quite naturally captured by the notion of order-sorted algebras [9], where the sort hierarchy of an order-sorted signature is induced by the inheritance relation of a given model and its function symbols represent the properties and queries specified in the model [10]. Following, the "states-as-algebras" paradigm [6], each order-sorted algebra represents a particular configuration of objects and links. In fact, we use order-sorted signatures, structures, and terms as substitutes for the precise OCL declarations in order to avoid some of its idiosyncrasies [3].

For expressiveness and ease of use, the OCL provides a set of built-in types, like `Boolean` or `Integer`, and collection constructors, like `Sequence` or `Set`, as well as a rich standard library. On the one hand, this library features primitive functions

for computations on values like c->including(e) for adding e to the sequence or set c. On the other hand, the construct c->iterate(i; a = e_0 | e) is available on collections which after initializing the accumulator variable a by e_0 successively binds the iteration variable i to the values in the collection c updating the accumulator with the result of evaluating e for the current values of i and a, and finally returns the value stored in a. Numerous operations on collections, like select, reject, or collect, but also forAll and exists, are built on top of this general iteration construct [5]. For accessing the currently available objects of a class allInstances() can be called on a type identifier; this call only succeeds when a type with finitely many inhabitants is used, such that Integer.allInstances() will not work.

In fact, OCL introduces a special value undefined for expressions like Integer.allInstances() or division by zero that do not yield a proper value. Instead of exception handling, the particular function isUndefined() can be used to check whether an expression results in undefined. The built-in Boolean functions and and or show a "parallel" (non-strict) behaviour for undefined, mandating that true or e and e or true always result in true, regardless of whether e yields undefined or not, and similarly for false and e and e and false.

We now consider these OCL features w.r.t. terms and evaluation one by one, but separately, starting with the order-sorted framework as a term charter and then accordingly adapting this framework. We restrict ourselves to an informal account of the notions mentioned above, that constitute the interesting cases within OCL. Formal, rigorous definitions can be found in the Appendix A.

4.1 Built-ins

The built-in types of OCL can be viewed as a particular case of the order-sorted framework in Ex. 1, namely the one that contains certain sorts and declarations and interprets them in the "standard" way. If we want, for instance, sequences and sets with membership test, then we require $\mathsf{Bool} \in |S|$ with $\{\mathsf{true}, \mathsf{false}\} \subseteq \delta_D(\mathsf{Bool})$, and $\{\mathsf{Seq}(s), \mathsf{Set}(s)\} \subseteq |S|$ with $-\!\!\rightarrow\!\mathsf{including}(-) \in \delta_D(\mathsf{Seq}(s)\ s, \mathsf{Seq}(s)) \cap \delta_D(\mathsf{Set}(s)\ s, \mathsf{Set}(s))$ (together with some sanity conditions). The morphisms are required to be the identity on these built-in types and function names. The signatures and morphisms fulfilling these requirements are called *primitives closed*, the sub-category they define is denoted by $\mathbb{S}^\circ$. Primitives-closed structures interpret built-in sorts and declarations in the standard way; this contravariant structure functor is denoted by Str°. Value domains, however, are not restricted: this means, in particular, that the value domain for $\mathsf{Set}(s)$ not necessarily consists of the (finite) sets of values in the value domain for s. The indexed category of values is thus the same as for order-sorted term charter, namely $Val^{\leq}$. The underlying indexed functor relating structures and values is denoted by U°. The terms are constructed in the same manner as those of the order-sorted case. This way, we obtain the *primitives-closed order-sorted term charter* $(\mathscr{C}^{\leq}, \nu^{\leq}, (-)^{\natural^{\leq}})$ over the term charter domain $(\mathbb{S}^\circ, Val^{\leq}, Str^\circ, U^\circ)$.

4.2 Iteration, All Instances, Undefinedness

The iteration construct of OCL is, in fact, a higher-order instrument since it binds both an iteration variable and an accumulator variable. Therefore, it cannot be treated as

the built-ins of above. It can however be added to primitive-closed term charters by including a further inductive case to the definition of the term language. Besides the base case of variables being a term and the inductive case of function symbols applied to previously defined terms, we have a second inductive case constructing an OCL iteration term: $t'{\to}\mathsf{iterate}(x'; x = t_0 \mid t)$ where t' is a term of collection type (with elements of type s'), t_0 is a term of arbitrary type s, x and x' are "new" variables of type s' and s, respectively, and t is a term of type s possibly containing x and x'. (For the sake of simplicity, we disregard here sub-sorting.) The extension of order-sorted signature morphisms to iteration terms is straightforward. The evaluation $(\beta)^{\natural^{\mathrm{it}}}$ along a valuation β is defined on iteration terms, if not in a straightforward, nevertheless in relatively simple manner by

$$
\begin{aligned}
&(\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(t'{\to}\mathsf{iterate}(x'; x = t_0 \mid t)) = \\
&\quad it((\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_{s'}(t'), (\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(t_0), \\
&\qquad \{(t_1, t_2) \mapsto ((\beta^{\mathrm{it}}\{x : s \mapsto t_2, x' : s' \mapsto t_1\})^{\natural^{\mathrm{it}}_M}_s(t)\})\}) \\
&\text{where } it(\varepsilon, t_a, f) = t_a \text{ and } it(t_i :: \ell, t_a, f) = it(\ell, f(t_i, t_a), f)
\end{aligned}
$$

The charter domain used here is the one of order-sorted signatures, that is, the obtained *iteration term charter* $(\mathscr{C}^{\mathrm{it}}, \nu^{\mathrm{it}}, (-)^{\natural^{\mathrm{it}}})$ is defined over the term charter domain $(\mathbb{S}^\circ, Val^{\leq}, Str^\circ, U^\circ)$.

Now, the introduction of the OCL query that returns all the instances of a given type, namely `allInstances`, conveys the introduction of an undefined return value if the type is infinite. Thus we consider a further special case of order-sorted value domains: those that contain the undefined constant †. More formally, the value domains remain unchanged, only the morphisms are "undef-lifted" and, in particular, the structures do not change, i.e., they do not contain †. This yields an indexed category $Val^\dagger : (\mathbb{S}^{\leq})^{\mathrm{op}} \to \mathrm{Cat}$ and thus an indexed functor $U^\dagger : Str^\circ \dot{\to} Val^\dagger$. Similarly as for iteration, a further inductive case is added to the definition of term language, namely $s.\mathsf{allInstances}()$ with s a sort. The extension of order-sorted signature morphisms as well as the (strict) extension of valuations to allInstances is straightforward:

$$
(\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_{s'}(s.\mathsf{allInstances}()) = \begin{cases} V_s & \text{if } |V_s| < \infty \\ \dagger & \text{otherwise} \end{cases}
$$

and in any other case the extension of the valuation β is strict. An *all-instances term charter* $(\mathscr{C}^{\mathrm{a}}, \nu^{\mathrm{a}}, (-)^{\natural^{\mathrm{a}}})$ over the term charter domain $(\mathbb{S}^\circ, Val^\dagger, Str^\circ, U^\dagger)$ is obtained.

Having a way to treat undefinedness of allInstances, the possibility of treating undefinedness in general opens up. So, for instance, non-strict functions as, e.g., if-then-else can be terms of the language. Three-valued Boolean connectives, moreover, need be defined. Again, not function symbols are assumed but further cases to the inductive definition of terms are added; in particular, the constant undef, the term construction $t.\mathsf{isUndef}()$ for t a term, t_1 and t_2 and t_1 or t_2 are terms if t_1 and t_2 are terms of sort Bool, and if t then t_1 else t_2 endif is a term if t is a term of sort Bool and t_1 and t_2 are of the same sort (disregarding sub-sorting here for the sake of simplicity). Both undef-lifted order-sorted signature morphisms and valuations are customarily defined on these new terms, with valuations strict but for if-then-else:

$$(\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(v_1 \text{ and } v_2) = \begin{cases} tt & \text{if } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_{\mathsf{Bool}}(v_1) = tt \text{ and } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_{\mathsf{Bool}}(v_2) = tt \\ ff & \text{if } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_{\mathsf{Bool}}(v_1) = ff \text{ or } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_{\mathsf{Bool}}(v_2) = ff \\ \dagger & \text{otherwise} \end{cases}$$

$$(\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(\text{if } v \text{ then } v_1 \text{ else } v_2 \text{ endif}) = \begin{cases} (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(v_1) & \text{if } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(v) = tt \\ (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(v_2) & \text{if } (\beta^{\mathfrak{u}})^{\natural^{\mathfrak{u}}_M}_s(v) = ff \\ \dagger & \text{otherwise} \end{cases}$$

The *undefinedness term charter* $(\mathscr{C}^{\mathfrak{u}}, \nu^{\mathfrak{u}}, (-)^{\natural^{\mathfrak{u}}})$ is thus defined over the term charter domain $(\mathbb{S}^\circ, Val^\dagger, Str^\circ, U^\dagger)$, i.e., over the same term charter domain as "all instances".

4.3 Institutions for OCL Sub-languages

The term charters of the preceding sections use primitives-closed signatures and structures. From each of them, by Prop. 2, corresponding institutions can be constructed by instantiating $\mathcal{U}_{Val^\dagger}$ and $*$. One possible choice is $\mathcal{U}_{Val^\dagger}(\langle \Sigma, V \rangle) = V_{\mathsf{Bool}}$ and taking the semantic truth value $*$ to be tt. With this choice a term of type Bool evaluating to $\dagger$ is per se not "true". Due to the satisfaction condition, this evaluation is invariant under change of notation.

Example 2. Assume that equality is one of the built-ins considered in Sect. 4.1 and let us write $t_1 = t_2$ instead of $=(t_1, t_2)$. In the undefinedness term charter $\mathfrak{T}^{\mathfrak{u}}$ with $\Sigma \in |\mathbb{S}^\circ|$, $X \in |\mathcal{G}(Val^\dagger)(\Sigma)|$, $M \in |\mathcal{G}(Str^\circ)(\Sigma)|$, and $\beta : X \to \mathcal{G}(U^\dagger)(M)$, we have $(\beta)^{\natural^{\mathfrak{u}}}_M(\mathsf{undef} = \mathsf{true}) = \dagger$ and therefore $(M, \beta) \not\models^{\mathcal{U}_{Val^\dagger}}_{\mathfrak{T}^{\mathfrak{u}}, X} \mathsf{undef} = \mathsf{true}$. Similarly, $(\beta)^{\natural^{\mathfrak{u}}}_M(\mathsf{false} = \mathsf{true}) = ff$, and again $(M, \beta) \not\models^{\mathcal{U}_{Val^\dagger}}_{\mathfrak{T}^{\mathfrak{u}}, X} \mathsf{false} = \mathsf{true}$. □

5 Operators on Term Charters

Having provided a series of examples for term charters for various OCL features in isolation, we now want to combine these term charters and thus the OCL features to obtain a coherent OCL semantics out of which we can also form an institution. We provide two first operators, which, however, both currently assume that all the involved term charters are given over the same term charter domain.

By sequencing term charters we can stack construction functors and thus get a levelled combination of their terms. Consider for example the all-instances term charter $\mathfrak{T}^{\mathfrak{a}} = (\mathscr{C}^{\mathfrak{a}}, \nu^{\mathfrak{a}}, (-)^{\natural^{\mathfrak{a}}})$ and the undefinedness term charter $\mathfrak{T}^{\mathfrak{u}} = (\mathscr{C}^{\mathfrak{u}}, \nu^{\mathfrak{u}}, (-)^{\natural^{\mathfrak{u}}})$ of Sect. 4.2 which are both defined over $(\mathbb{S}^\circ, Val^\dagger, Str^\circ, U^\dagger)$. In the term charter $\mathfrak{T}^{\mathfrak{a}} \rhd \mathfrak{T}^{\mathfrak{u}} = (\mathscr{C}, \nu, (-)^\natural)$ resulting from sequencing these two term charters we obtain the "heterogeneous" term $(s.\mathsf{allInstances}()).\mathsf{isUndef}()$. This sequencing can be iterated thus adding more levels; a full combination, that allows the occurrence of terms from both term charters on all levels, is in a co-limit construction provided below of the chain $\nu_{\mathscr{C}^{(n)}} : \mathscr{C}^{(n)} \to \mathscr{C}^{(n+1)}$ where $\mathscr{C}^{(n)}$ is the construction functor of the n-th level.

Both operators, sequencing and co-limit, work in the category $\mathrm{TmCh}(\mathsf{D})$ of term charters over a given term charter domain D, where a *term charter morphism* $\mu : \mathfrak{T}_1 \to \mathfrak{T}_2$ with term charters $\mathfrak{T}_1 = (\mathscr{C}_1, \nu_1, (-)^{\natural_1})$ and $\mathfrak{T}_2 = (\mathscr{C}_2, \nu_2, (-)^{\natural_2})$ over $\mathsf{D} = (\mathbb{S}, Val, Str, U)$ is given by a natural transformation $\mu : \mathscr{C}_1 \dot{\to} \mathscr{C}_2$ such that for all $\Sigma \in |\mathbb{S}|$, $X \in |\mathcal{G}(Val)(\Sigma)|$, $M \in |\mathcal{G}(Str)(\Sigma)|$ and $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$ the conditions $\nu_{1,X}; \mu_X = \nu_{2,X}$ and $\mu_X; (\beta)^{\natural_{2,M}} = (\beta)^{\natural_{1,M}}$ hold.

5.1 Sequencing of Term Charters

Let $\mathfrak{T}_1 = (\mathscr{C}_1, \nu_1, (-)^{\natural_1})$ and $\mathfrak{T}_2 = (\mathscr{C}_2, \nu_2, (-)^{\natural_2})$ be term charters over the term charter domain $(\mathbb{S}, Val, Str, U)$. Then the *sequencing* $\mathfrak{T}_1 \rhd \mathfrak{T}_2 = (\mathscr{C}, \nu, (-)^{\natural})$ of first $\mathfrak{T}_1$ and then $\mathfrak{T}_2$ is defined by

$$\begin{aligned}
&\mathscr{C} = \mathscr{C}_1; \mathscr{C}_2 : \mathcal{G}(Val) \to \mathcal{G}(Val)\\
&\nu_X = \nu_{1,X}; \nu_{2,\mathscr{C}_1(X)} = \nu_{2,X}; \mathscr{C}_2(\nu_{1,X}) : X \to \mathscr{C}_2(\mathscr{C}_1(X))\\
&\beta^{\natural_M} = (\beta^{\natural_{1,M}})^{\natural_{2,M}}
\end{aligned}$$

for all $X \in |\mathcal{G}(Val)(\Sigma)|$, $M \in |\mathcal{G}(Str)(\Sigma)|$, and $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$.

Proposition 3. *Let* $\mathsf{D} = (\mathbb{S}, Val, Str, U)$ *be a term charter domain. Let* $\mathfrak{T}_1 = (\mathscr{C}_1, \nu_1, (-)^{\natural_1})$ *and* $\mathfrak{T}_2 = (\mathscr{C}_2, \nu_2, (-)^{\natural_2})$ *be term charters over* D*. Then* $\mathfrak{T}_1 \rhd \mathfrak{T}_2$ *is a term charter over* D*.* □

Example 3. Consider the "heterogeneous" term $(\mathsf{Integer.allInstances}()).\mathsf{isUndef}()$ of $\mathfrak{T}^{\mathrm{a}} \rhd \mathfrak{T}^{\mathrm{u}}$ where we assume that $\mathsf{Integer}$ is a built-in sort standardly interpreted by $\mathbb{Z}$. This term is built by first constructing $\mathsf{Integer.allInstances}()$ in $\mathfrak{T}^{\mathrm{a}}$, then taking this term as a variable, which we may abbreviate by x, and constructing $x.\mathsf{isUndef}()$ in $\mathfrak{T}^{\mathrm{u}}$. Consequently, the evaluation of

$$((\beta)^{\natural^{\mathrm{a}}}_M)^{\natural^{\mathrm{u}}}_M((\mathsf{Integer.allInstances}()).\mathsf{isUndef}()) = ((\beta)^{\natural^{\mathrm{a}}}_M)^{\natural^{\mathrm{u}}}_M(x.\mathsf{isUndef}())$$

for an arbitrary $\beta : X \to \mathcal{G}(U^{\dagger})(M)$ with $X \in |\mathcal{G}(Val^{\dagger})(\Sigma)|$, $M \in |\mathcal{G}(Str^{\circ})(\Sigma)|$, and $\Sigma \in |\mathbb{S}^{\circ}|$ first evaluates $((\beta)^{\natural^{\mathrm{a}}}_M)^{\natural^{\mathrm{u}}}_M(x)$, amounting to $(\beta)^{\natural^{\mathrm{a}}}_M(x)$, since x is a variable, which yields $\dagger$. Thus the overall result is tt. □

Also the natural transformation $\nu_{2,\mathscr{C}_1(-)} : \mathscr{C}_1 \dot{\to} \mathscr{C}_1; \mathscr{C}_2$ induces a term charter morphism from $\mathfrak{T}_1$ to $\mathfrak{T}_1 \rhd \mathfrak{T}_2$, and, likewise, the natural transformation $\mathscr{C}_2(\nu_1) : \mathscr{C}_2 \dot{\to} \mathscr{C}_1; \mathscr{C}_2$ induces a term charter morphism from $\mathfrak{T}_2$ to $\mathfrak{T}_1 \rhd \mathfrak{T}_2$. The *$n$-th iteration* $\mathfrak{T}^{(n)}$ of a term charter $\mathfrak{T}$ for $n \geq 1$ is inductively defined by $\mathfrak{T}^{(1)} = \mathfrak{T}$ and $\mathfrak{T}^{(n+1)} = \mathfrak{T}^{(n)} \rhd \mathfrak{T}$.

5.2 Co-limits of Term Charters

Let $\mathsf{D} = (\Sigma, Val, Str, U)$ be a term charter domain. For a term charter $\mathfrak{T} = (\mathscr{C}, \nu, (-)^{\natural}) \in |\mathrm{TmCh}(\mathsf{D})|$ let us write $\mathfrak{T}_{\mathscr{C}}$, $\mathfrak{T}_{\nu}$, and $\mathfrak{T}_{\natural}$ for the components of $\mathfrak{T}$. Consider a diagram $F : J \to \mathrm{TmCh}(\mathsf{D})$ where J is a small connected category. Assume that for every $X \in |\mathcal{G}(Val)(\Sigma)|$ with $\Sigma \in |\mathbb{S}|$, the diagram $F_{\mathscr{C},X} : J \to \mathcal{G}(Val)(\Sigma)$

with $F_{\mathscr{C},X}(j) = F(j)_{\mathscr{C}}(X)$ and $F_{\mathscr{C},X}(f : j \to j') = F(f)_X$ has co-limit $(C_{F,X} \in \mathcal{G}(Val)(\Sigma), \gamma_{F,X} : F_{\mathscr{C},X} \stackrel{\cdot}{\to} \Delta(C_{F,X}))$ (where, for a category $\mathbb{C}$, $\Delta : \mathbb{C} \to \mathbb{C}^J$ denotes the diagonal functor mapping a $C \in |\mathbb{C}|$ to the functor $\Delta(C) : J \to \mathbb{C}$ with $\Delta(C)(j) = C$ and $\Delta(C)(f : j \to j') = 1_C$). Then, by universality, for each $\xi : X \to Y$ in $\mathcal{G}(Val)(\Sigma)$ there is a unique arrow $c_{F,\xi} : C_{F,X} \to C_{F,Y}$ such that

$$F(j)_{\mathscr{C}}(\xi); \gamma_{F,Y,j} = \gamma_{F,X,j}; c_{F,\xi} \quad \text{for all } j \in |J| \,.$$

Define $\mathscr{C}^F(X) = C_{F,X}$ and $\mathscr{C}^F(\xi) = c_{F,\xi}$. Furthermore, for all $f : j \to j'$ in J,

$$(F(j)_\nu)_X; F(f)_X = (F(j')_\nu)_X \quad \text{and} \quad \gamma_{F,X,j} = F(f)_X; \gamma_{F,X,j'}$$

Define $\nu^F_X = (F(j)_\nu)_X; \zeta_{F,X,j}$ for some $j \in |J|$. For a morphism $\beta : X \to \mathcal{G}(U)(M)$ in $\mathcal{G}(Val)(\Sigma)$ with $M \in |\mathcal{G}(Str)(\Sigma)|$ let $(\beta)^{\natural^F_M} : \mathscr{C}^F(X) \to \mathcal{G}(U)(M)$ be the unique morphism with $\zeta_{F,X,j}; (\beta)^{\natural^F_M} = F(j)_\natural(\beta)$ for all $j \in |J|$ which exists since

$$F(f)_X; F(j)_{\natural_M}(\beta) = F(j')_{\natural_M}(\beta) \quad \text{for all } f : j \to j' \text{ in } J.$$

Then $\mathfrak{T}^F = (\mathscr{C}^F, \nu^F, (-)^{\natural^F})$ is a term charter and all $\gamma_{F,-,j}$ are term charter morphisms. In fact, $(\mathfrak{T}^F, \gamma^F)$ with $(\gamma^F_j)_X = \gamma_{F,X,j}$ is the co-limit of F.

Proposition 4. $(-)_{\mathscr{C}} : \mathrm{TmCh}(\mathbb{S}, Val, Str, U) \to \mathrm{Fun}(\mathcal{G}(Val), \mathcal{G}(Val))$ *creates parameterized small connected co-limits.* □

Example 4. Continuing the previous example, we now want to consider arbitrarily nested terms from the all-instances term charter $\mathfrak{T}^{\mathrm{a}}$ and the undefinedness term charter $\mathfrak{T}^{\mathrm{u}}$. We thus consider the chain $\mathfrak{T} \stackrel{\nu_1}{\longrightarrow} \mathfrak{T}^{(2)} \stackrel{\nu_2}{\longrightarrow} \mathfrak{T}^{(3)} \stackrel{\nu_3}{\longrightarrow} \cdots$ 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 $\mathscr{C}(X) \stackrel{\nu_{1,X}}{\longrightarrow} \mathscr{C}^{(2)}(X) \stackrel{\nu_{2,X}}{\longrightarrow} \mathscr{C}^{(3)}(X) \stackrel{\nu_{3,X}}{\longrightarrow} \cdots$ has a co-limit in $\mathcal{G}(Val^\dagger)(\Sigma)$ for $X \in |\mathcal{G}(Val^\dagger)(\Sigma)|$. Indeed, the co-limit object of this chain is simply given by the component-wise union of the value domains and thus we obtain a co-limit term charter by Prop. 4. The evaluation of a term at a nesting level n of $\mathfrak{T}^{\mathrm{a}}$ and $\mathfrak{T}^{\mathrm{u}}$ then proceeds like in $\mathfrak{T}^{(n)}$. □

6 Conclusions and Future Work

Along the lines of Martin Wirsing's proposal for the definition of an heterogeneous semantics of UML in [4], we have presented above a semantics for the constraints language OCL. The distinct characteristic of this approach is the compositional construction of theories out of basic ones. Indeed, OCL can be obtained by sequencing term charters and building the co-limit of the result. This way only the theories, presented as term charters and needed for the situation at hand, are combined into an OCL sub-language (and therefore the theories that are dispensable need not be included).

Let us emphasize that OCL is not a logic but a term language. It it is imperative to deal with its particularities, too, especially undefinedness and non-termination. The OCL setting defines a logic that is not binary. That is, we have to deal with formulase

that, instead of being either true or not, may be true, false, undefined or even non-terminating. In order to mimic the implied three- or four-valued OCL logic (see [3] and also [12]), the most natural to do is, on the one hand, to follow the definition of OCL as closely as possible and construct a term language whose equality, on the other hand, is invariant under change of notation. As pointed out above, we moreover addressed the OCL sub-languages one by one, thus supporting compositional construction of term languages that comply the satisfaction condition and can consequently be presented as institutions.

Indeed, term charters for the OCL sub-languages can be composed by means of the sequencing operator and the co-limit construction of Sect. 5, provided they are defined over the same term charter domain. In particular, an OCL term charter can be obtained by composing the term charters sketched in Sect. 4, that is, the primitives-closed order-sorted term charter (see Sect. A.2), the iteration term charter (see Sect. A.3), the all-instances term charter (see Sect. A.4), and the undefinedness term charter (see Sect. A.5), only after their reformulation as term charters over a single term charter domain. By Prop. 2, the resulting OCL term charter defines an institution for OCL.

Useful would be the possibility of combining term charters defined over different term charter domains. The use of heterogeneous term charter domains could support the construction of the four-valued logic with undef and non-termination by means of operators on the corresponding three-valued term charters, i.e., by composing them directly, instead of resorting to their redefinition for a four-valued term charter domain. A property not demonstrated yet is the associativity of sequencing. A further issue, to be included in the framework presented in this work, is the treatment of pre-/post-conditions. The present idea consists in testing them on pairs of "states", one that represents the state at the time before the execution of a method and one that represents the state afterwards. In the long term, we aim to define an entailment system for OCL which, combined with the institution above, would yield an general logic; see [13]. Finally, an integration into the institution-based Heterogeneous Tool Set [14] for analysis and proof support in multi-logic specifications is planned.

Acknowledgements. We thank Till Mossakowski and Hubert Baumeister for fruitful discussions and comments on previous drafts of this work. We especially want to express our deep gratitude to Martin Wirsing for initiating this work, accompanying our scientific lives, and particularly for being, both from the academic and personal point of view, precisely Martin Wirsing.

References

1. Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Addison-Wesley (1999)
2. Boronat, A., Knapp, A., Meseguer, J., Wirsing, M.: What Is a Multi-modeling Language? In: Corradini, A., Montanari, U. (eds.) WADT 2008. LNCS, vol. 5486, pp. 71–87. Springer, Heidelberg (2009)
3. Cengarle, M.V., Knapp, A.: OCL 1.4/1.5 vs. OCL 2.0 Expressions: Formal Semantics and Expressiveness. Softw. Syst. Model. 3(1), 9–30 (2004)

4. Cengarle, M.V., Knapp, A., Tarlecki, A., Wirsing, M.: A Heterogeneous Approach to UML Semantics. In: Degano, P., De Nicola, R., Meseguer, J. (eds.) Concurrency, Graphs and Models. LNCS, vol. 5065, pp. 383–402. Springer, Heidelberg (2008)
5. Clark, T.: Typechecking UML Static Models. In: France, R.B. (ed.) UML 1999. LNCS, vol. 1723, pp. 503–517. Springer, Heidelberg (1999)
6. Ganzinger, H.: Programs as Transformations of Algebraic Theories (Extended Abstract). Informatik Fachberichte 50, 22–41 (1981)
7. Goguen, J.A., Burstall, R.M.: A Study in the Foundation of Programming Methodology: Specifications, Institutions, Charters, and Parchments. In: Poigné, A., Pitt, D.H., Rydeheard, D.E., Abramsky, S. (eds.) Category Theory and Computer Programming. LNCS, vol. 240, pp. 313–333. Springer, Heidelberg (1986)
8. Goguen, J.A., Burstall, R.M.: Institutions: Abstract Model Theory for Specification and Programming. J. ACM 39(1), 95–146 (1992)
9. Goguen, J.A., Meseguer, J.: Order-sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. Theo. Comp. Sci. 105(2), 217–273 (1992)
10. Hennicker, R., Knapp, A., Baumeister, H.: Semantics of OCL Operation Specifications. In: Schmitt, P.H. (ed.) Proc. Wsh. OCL 2.0 — Industry Standard or Scientific Playground? (WOCL 2003). Electr. Notes Theo. Comp. Sci., vol. 120, pp. 111–132. Elsevier (2004)
11. Knapp, A., Cengarle, M.V.: Institutions for OCL-like Expression Languages. Manuscript, Universitt Augsburg (2014), `http://www.informatik.uni-augsburg.de/lehrstuehle/swt/sse/veroeffentlichungen/uau-2014/ocl-institutions.pdf`
12. Lano, K.: Null Considered Harmful (for Transformation Verification). In: Proc. 3rd Int. Wsh. Verification of Model Transformations, VOLT 2014 (2014), `http://volt2014.big.tuwien.ac.at/papers/volt2014_paper_3.pdf`
13. Meseguer, J.: General Logics. In: Ebbinghaus, H.D., Fernández-Prida, J., Garrido, M., Lascar, D., Rodríguez Artalejo, M. (eds.) Proc. Logic Colloquium 1987, pp. 275–329. North-Holland (1989)
14. Mossakowski, T., Maeder, C., Lüttich, K.: The Heterogeneous Tool Set, HETS. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 519–522. Springer, Heidelberg (2007)
15. Object Management Group: Unified Modeling Language, Superstructure. Version 2.4.1. Specification formal/2011-08-06, OMG (2011)
16. Object Management Group: Object Constraint Language. Version 2.3.1. Specification formal/2012-01-01, OMG (2012)
17. Pawłowski, W.: Context Parchments. In: Parisi-Presicce, F. (ed.) WADT 1997. LNCS, vol. 1376, pp. 381–401. Springer, Heidelberg (1998)
18. Petre, M.: UML in Practice. In: Proc. 35th Int. Conf. Software Engineering (ICSE 2013), pp. 722–731. IEEE (2013)
19. Sannella, D., Tarlecki, A.: Foundations of Algebraic Specification and Formal Software Development. EATCS Monographs in Theoretical Computer Science. Springer (2012)
20. Tarlecki, A., Burstall, R.M., Goguen, J.A.: Some Fundamental Algebraic Tools for the Semantics of Computation, Part 3: Indexed Categories. Theo. Comp. Sci. 91, 239–264 (1991)
21. Wirsing, M.: Algebraic Specification. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics, pp. 675–788. Elsevier and MIT Press (1990)

A OCL Terms and Evaluation

We formally define the OCL features discussed in Sect. 4 following the same strategy, i.e., one by one and each time adapting the framework.

A.1 Order-Sorted Terms and Evaluation

For the first basic step, we recapitulate the notions of order-sorted signatures, structures, terms, and evaluation in terms of indexed categories.

Signatures, Values, and Structures. An *order-sorted signature* (S, D) consists of a *sort hierarchy* S and a *function declaration* pair over S; where $S = (|S|, \leq_S)$ is a partial order with a set of sort names $|S|$ and a sub-sorting relation $\leq_S$; and $D = (|D|, \delta_D)$ is a pair with $|D|$ a set of function names and $\delta_D : |S|^* \times |S| \to \mathcal{P}(|D|)$ a function such that $|D| = \bigcup\{\delta_D(\overline{s}, s) \mid \overline{s} \in |S|^*, s \in |S|\}$. An *order-sorted signature morphism* $(\gamma, \rho) : (S, D) \to (S', D')$ is given by a monotone function $\gamma : S \to S'$ and a function $\rho : |D| \to |D'|$ such that $\rho(d) \in \delta_{D'}(\gamma(\overline{s}), \gamma(s))$ for each $d \in \delta_D(\overline{s}, s)$. Order-sorted signatures and morphisms between them define a category which we denote by $\mathbb{S}^{\leq}$.

An (S, D)-*value domain* V consists of a family $V = (V_s)_{s \in |S|}$ of sets of *values* with $V_s \subseteq V_{s'}$ if $s \leq_S s'$. An (S, D)-*value domain morphism* $\omega : V \to V'$ is given by a family of mappings $\omega = (\omega_s : V_s \to V'_s)_{s \in |S|}$. (S, D)-value domains and morphisms define a category $Val^{\leq}(S, D)$. The indexed category $Val^{\leq} : (\mathbb{S}^{\leq})^{\mathrm{op}} \to \mathrm{Cat}$ maps each $\Sigma = (S, D)$ to $Val^{\leq}(\Sigma)$ and each $(\gamma, \rho) : \Sigma \to \Sigma' = (S', D')$ to the functor $Val^{\leq}(\gamma, \rho) : Val^{\leq}(\Sigma') \to Val^{\leq}(\Sigma)$ with $Val^{\leq}(\gamma, \rho)((V'_{s'})_{s' \in |S'|}) = (V'_{\gamma(s)})_{s \in |S|}$ and $Val^{\leq}(\gamma, \rho)((\omega'_{s'} : V'_{1,s'} \to V'_{2,s'})_{s' \in |S'|}) = (\omega'_{\gamma(s)} : V'_{1,\gamma(s)} \to V'_{2,\gamma(s)})_{s \in |S|}$.

An (S, D)-*structure* (V, E) consists of an (S, D)-value domain and a family of *evaluation* functions $E = (E_{\overline{s},s})_{\overline{s} \in |S|^*, s \in |S|}$ with $E_{\overline{s},s} : \delta_D(\overline{s}, s) \to (V_{\overline{s}} \to V_s)$. An (S, D)-*structure morphism* $\omega : (V, E) \to (V', E')$ is given by an (S, D)-value domain morphism $\omega : V \to V'$ such that the *homomorphism condition* $\omega_s(E_{\overline{s},s}(d)(\vec{v})) = E'_{\overline{s},s}(d)(\omega_{\overline{s}}(\vec{v}))$ is satisfied. (S, D)-structures and morphisms define a category $Str^{\leq}(S, D)$. The indexed category $Str^{\leq} : (\mathbb{S}^{\leq})^{\mathrm{op}} \to \mathrm{Cat}$ maps each Σ to $Str^{\leq}(\Sigma)$ and each $(\gamma, \rho) : \Sigma \to \Sigma' = (S', D')$ to the functor $Str^{\leq}(\gamma, \rho) : Str^{\leq}(\Sigma') \to Str^{\leq}(\Sigma)$ with $Str^{\leq}(\gamma, \rho)((V'_{s'})_{s' \in |S'|}, (E'_{\overline{s}',s'})_{\overline{s}' \in |S'|^*, s' \in |S'|}) = ((V'_{\gamma(s)})_{s \in |S|}, ((E' \circ \rho)_{\gamma(\overline{s}),\gamma(s)})_{\overline{s} \in |S|^*, s \in |S|})$ and $Str^{\leq}(\gamma, \rho)(\omega') = Val^{\leq}(\gamma, \rho)(\omega')$.

The indexed functor $U^{\leq} : Str^{\leq} \dot{\to} Val^{\leq}$ "forgets" the evaluation functions of a structure.

Terms and Evaluation. For constructing *order-sorted terms* over an order-sorted value domain we define a functor $\mathscr{C}^{\leq} : \mathcal{G}(Val^{\leq}) \to \mathcal{G}(Val^{\leq})$ as follows: For an object $\langle \Sigma, X \rangle \in |\mathcal{G}(Val^{\leq})|$ with $\Sigma = (S, D)$ set $\mathscr{C}^{\leq}(\langle \Sigma, X \rangle) = \langle \Sigma, V^{\leq}_X \rangle$ such that for each $s \in |S|$ the values in $V^{\leq}_s$ are given inductively by

- $x \in V^{\leq}_{X,s}$ for $x \in X_s$;
- $d(\vec{v}) \in V^{\leq}_{X,s'}$ for all $s' \geq_S s$ if $d \in \delta_D(\overline{s}, s)$ and $\vec{v} \in V^{\leq}_{X,\overline{s}}$;

For a morphism $\langle \sigma, \omega \rangle : \langle \Sigma, X \rangle \to \langle \Sigma', X' \rangle$ in $\mathcal{G}(Val^{\leq})$ with $\sigma = (\gamma, \rho)$ and $\Sigma = (S, D)$ set $\mathscr{C}^{\leq}(\langle \sigma, \omega \rangle) = \langle \sigma, \omega^{\leq} \rangle : \mathscr{C}^{\leq}(\langle \Sigma, X \rangle) \to \mathscr{C}^{\leq}(\langle \Sigma', X' \rangle)$ such that inductively $\omega^{\leq}_s(x) = \omega_s(x)$ for $x \in X_s$ and $\omega^{\leq}_s(d(\vec{v})) = \rho(d)(\omega^{\leq}_{\overline{s}}(\vec{v}))$.

For *evaluating* order-sorted terms over a Σ-structure $M = \langle \Sigma, (V, E) \rangle$ in $|\mathcal{G}(Str^{\leq})|$ given a *valuation* $\beta = \langle 1_\Sigma, \beta^{\leq} \rangle : \langle \Sigma, X \rangle \to \mathcal{G}(U^{\leq})(\langle \Sigma, (V, E) \rangle)$ define $(\beta)^{\natural^{\leq}_M} = \langle 1_\Sigma, ((\beta^{\leq}))^{\natural^{\leq}_M} \rangle : \mathscr{C}^{\leq}(\langle \Sigma, X \rangle) \to \langle \Sigma, (V, E) \rangle$ inductively by

- $(\beta^{\leq})^{\natural^{\leq}_M}_s(x) = \beta^{\leq}_s(x)$ for $x \in X_s$;
- $(\beta^{\leq})^{\natural^{\leq}_M}_s(d(\vec{v})) = E_{\overline{s},s}(d)((\beta^{\leq})^{\natural^{\leq}_M}_{\overline{s}}(\vec{v}))$.

Term Charter. Given an order-sorted signature morphism $\sigma : \Sigma \to \Sigma'$ in $\mathbb{S}^{\leq}$, a Σ'-structure $\langle \Sigma', (V', E') \rangle \in |\mathcal{G}(Str^{\leq})(\Sigma')|$, and a valuation $\beta' : \langle \Sigma', X' \rangle \to \mathcal{G}(U^{\leq})(\langle \Sigma', (V', E') \rangle)$ in $\mathcal{G}(Val^{\leq})(\Sigma')$ it is straightforwardly checked that the evaluation condition (E) for term charters

$$\begin{array}{ccc}
\mathscr{C}^{\leq}(\langle \Sigma', X' \rangle|_{Val^{\leq}\sigma}) & \xrightarrow{\mathscr{C}^{\leq}(\sigma|_{Val^{\leq}}\langle \Sigma', X' \rangle)} & \mathscr{C}^{\leq}(\langle \Sigma', X' \rangle) \\
\downarrow{\scriptstyle (\beta'|_{Val^{\leq}\sigma})^{\natural^{\leq}_{\langle \Sigma', (V', E') \rangle|_{Str^{\leq}\sigma}}}} & & \downarrow{\scriptstyle (\beta')^{\natural^{\leq}_{\langle \Sigma', (V', E') \rangle}}} \\
\mathcal{G}(U^{\leq})(\langle \Sigma', (V', E') \rangle|_{Str^{\leq}\sigma}) & \xrightarrow[\mathcal{G}(U^{\leq})(\sigma|_{Str^{\leq}}\langle \Sigma', (V', E') \rangle)]{} & \mathcal{G}(U^{\leq})(\langle \Sigma', (V', E') \rangle)
\end{array}$$

indeed is satisfied. Also condition (K) is easily shown. As natural transformation $\nu^{\leq} : 1_{\mathcal{G}(Val^{\leq})} \dot{\to} \mathscr{C}^{\leq}$ for embedding values or variables into the order-sorted terms we may simply choose the inclusions, i.e., $\nu^{\leq}_{\langle (S,D), X \rangle} = \langle 1_{(S,D)}, (\iota_{\langle (S,D), X \rangle, s} : X_s \to V^{\leq}_{X,s})_{s \in |S|} \rangle$ which also satisfies (C).

Thus we obtain the *order-sorted term charter* $(\mathscr{C}^{\leq}, \nu^{\leq}, (-)^{\natural^{\leq}})$ over the term charter domain $(\mathbb{S}^{\leq}, Val^{\leq}, Str^{\leq}, U^{\leq})$.

A.2 Adding Built-ins

The addition of OCL's built-in types can be handled by a specialization of order-sorted signatures and structures, requiring them to contain and interpret particular sorts and declarations in a standard way. We demonstrate this by adding Booleans, sequences, and sets as well as a few functions; these additions are by far not exhaustive, but meant to be exemplarily. Nevertheless, we call the resulting order-sorted signatures and structures "primitives closed".

Signatures and Structures. An order-sorted signature (S, D) with $S = (|S|, \leq_S)$ and $D = (|D|, \delta_D)$ is *primitives-closed* whenever $\mathsf{Bool} \in |S|$, and $\mathsf{true} \in \delta_D(\mathsf{Bool})$ and $\mathsf{false} \in \delta_D(\mathsf{Bool})$; and the following conditions hold for all $\tau \in \{\mathsf{Seq}, \mathsf{Set}\}$ and all $s, s' \in |S|$:

- $\tau(s) \in |S|$ if, and only if, $s \in |S|$;
- $\tau(s) \leq_S \tau(s')$ if, and only if, $s \leq_S s'$;
- $\tau\{\} \in \delta_D(\tau(s))$ and $-{\to}\mathsf{including}(-) \in \delta_D(\tau(s)\ s, \tau(s))$.

A morphism $(\gamma, \rho) : (S, D) \to (S', D')$ between primitives-closed order-sorted signatures is *primitives-closed* if $\gamma(\mathsf{Bool}) = \mathsf{Bool}$, and $\rho(\mathsf{true}) = \mathsf{true}$ and $\rho(\mathsf{false}) = \mathsf{false}$; and the following conditions hold for all $\tau \in \{\mathsf{Seq}, \mathsf{Set}\}$ and all $s \in |S|$:

- $\gamma(\tau(s)) = \tau(\gamma(s))$;
- $\rho(\tau\{\}) = \tau\{\}$ and $\rho(-\!\!\to\!\mathsf{including}(-)) = -\!\!\to\!\mathsf{including}(-)$.

Let $\mathbb{S}^\circ$ be the sub-category of order-sorted signatures consisting of all the primitives-closed order-sorted signatures and all the primitives-closed morphisms between them.

An (S, D)-structure (V, E) over a primitives-closed order-sorted signature (S, D) is *primitives-closed* if $V_{\mathsf{Bool}} = \{tt, \mathit{ff}\}$, and $E_{\mathsf{Bool}}(\mathsf{true}) = tt$ and $E_{\mathsf{Bool}}(\mathsf{false}) = \mathit{ff}$; and for all $s \in |S|$:

- $V_{\mathsf{Seq}(s)} = (V_s)^*$ and $V_{\mathsf{Set}(s)} = \mathcal{P}_{\mathrm{fin}}(V_s)$ (i.e., all finite lists and sets over V_s);
- $E_{\mathsf{Seq}(s)}(\mathsf{Seq}\{\}) = \varepsilon$ and $E_{\mathsf{Set}(s)}(\mathsf{Set}\{\}) = \emptyset$ (i.e., the empty list and set);
- $E_{\mathsf{Seq}(s)\ s,\mathsf{Seq}(s)}(-\!\!\to\!\mathsf{including}(-)) = \{(l, v) \mapsto v :: l\}$ (i.e., prepending an element to a list) and $E_{\mathsf{Set}(s)\ s,\mathsf{Set}(s)}(-\!\!\to\!\mathsf{including}(-)) = \{(m, v) \mapsto \{v\} \cup m\}$ (i.e., adding an element to a set);

An (S, D)-structure morphism $\omega : (V, E) \to (V', E')$ over a primitives-closed order-sorted signature (S, D) is *primitives-closed* if $\omega_{\mathsf{Bool}}(tt) = tt$ and $\omega_{\mathsf{Bool}}(\mathit{ff}) = \mathit{ff}$; and for all $s \in |S|$:

- $\omega_{\mathsf{Seq}(s)}(\varepsilon) = \varepsilon$ and $\omega_{\mathsf{Set}(s)}(\emptyset) = \emptyset$;
- $\omega_{\mathsf{Seq}(s)}(v :: l) = \omega_s(v) :: \omega_{\mathsf{Seq}(s)}(l)$ and $\omega_{\mathsf{Set}(s)}(\{v\} \cup m) = \{\omega_s(v)\} \cup \omega_{\mathsf{Set}(s)}(m)$.

The indexed category $\mathit{Str}^\circ : (\mathbb{S}^\circ)^{\mathrm{op}} \to \mathrm{Cat}$ is defined like $\mathit{Str}^{\leq}$ but only involves primitives-closed order-sorted signatures, structures, and morphisms.

The indexed functor $U^\circ : \mathit{Str}^\circ \dot{\to} \mathit{Val}^{\leq}$ is defined by $U^{\leq}$ restricted to Str°.

Terms and Evaluation. As construction functor for *primitives-closed* order-sorted terms we can still use $\mathscr{C}^{\leq} : \mathcal{G}(\mathit{Val}^{\leq}) \to \mathcal{G}(\mathit{Val}^{\leq})$ as defined in Sect. A.1. Also the definition of the evaluation of primitives-closed order-sorted terms, though now involving primitives-closed signatures and structures, stays the same, such that the corresponding evaluation condition (E) again is satisfied. However, the primitives-closed order-sorted terms do not directly give rise to primitives-closed structures (in the sense of term algebras) due to the "standard interpretation" requirements on sequences and sets.

Term Charter. In particular, we obtain the *primitives-closed order-sorted term charter* $(\mathscr{C}^{\leq}, \nu^{\leq}, (-)^{\natural^{\leq}})$ over the term charter domain $(\mathbb{S}^\circ, \mathit{Val}^{\leq}, \mathit{Str}^\circ, U^\circ)$. Furthermore, setting $\mathcal{U}_{\mathit{Val}^{\leq}}(\langle \Sigma, V \rangle) = V_{\mathsf{Bool}}$ and $* = tt$ we obtain, by applying Prop. 2, an institution for the primitives-closed order-sorted term charter. Since $\mathit{Val}^{\leq}(\Sigma)$ has initial value domains for each $\Sigma \in |\mathbb{S}^\circ|$, we can also apply Cor. 1 and obtain a "classical" institution.

A.3 Iteration

For handling OCL's iteration construct, we only extend the term language, but keep working over primitives-closed order-sorted signatures and structures. In fact, iteration is not straightforwardly integrable into order-sorted signatures and structures themselves, since it binds the iteration and the accumulator variable and thus involves higher-order terms.

Terms and Evaluation. The construction functor for *iteration terms* $\mathscr{C}^{\mathrm{it}} : \mathcal{G}(Val^{\leq}) \to \mathcal{G}(Val^{\leq})$ is defined as follows: For the objects, set $\mathscr{C}^{\mathrm{it}}(\langle (S,D), X\rangle) = \langle (S,D), V^{\mathrm{it}}_X\rangle$ such that inductively

- $x \in V^{\mathrm{it}}_{X,s}$ if $x \in X_s$;
- $d(\vec{v}) \in V^{\mathrm{it}}_{X,s'}$ for all $s' \geq_S s$ if $d \in \delta_D(\overline{s}, s)$ and $\vec{v} \in V^{\mathrm{it}}_{X,\overline{s}}$;
- $v' \to \mathsf{iterate}(y' ; y = v_0 \,|\, v) \in V^{\mathrm{it}}_{X,s}$ if $v' \in V^{\mathrm{it}}_{X,\mathsf{Seq}(s')}$ with $s' \in |S|$, $v_0 \in V^{\mathrm{it}}_{X,s}$, and $v \in V^{\mathrm{it}}_{X \uplus \{y:s,y':s'\},s}$ (where, for $s_0, s_1 \in |S|$, $y_0 \notin X_{s'_0}$ for any $s'_0 \geq_S s_0$, $(X \uplus \{y_0 : s_0\})_{s_1}$ is defined by X_{s_1} if $s_0 \not\leq_S s_1$ and by $X_{s_1} \cup \{y_0\}$ if $s_0 \leq_S s_1$).

For the morphisms, define the morphism $\mathscr{C}^{\mathrm{it}}(\langle (\gamma,\rho), \omega\rangle) = \langle (\gamma,\rho), \omega^{\mathrm{it}}\rangle : \mathscr{C}^{\mathrm{it}}(\langle (S,D), X\rangle) \to \mathscr{C}^{\mathrm{it}}(\langle (S',D'), X'\rangle)$ such that, by simultaneous induction, $\omega^{\mathrm{it}}_s(x) = \omega_s(x)$ for $x \in X_s$; $\omega^{\mathrm{it}}_s(d(\vec{v})) = \rho(d)(\omega^{\mathrm{it}}_{\overline{s}}(\vec{v}))$; and $\omega^{\mathrm{it}}_s(v' \to \mathsf{iterate}(y' ; y = v_0 \,|\, v)) = \omega^{\mathrm{it}}_{s'}(v') \to \mathsf{iterate}(y' ; y = \omega^{\mathrm{it}}_s(v_0) \,|\, (\omega\{y : s \mapsto y : \gamma(s), y' : s' \mapsto y' : \gamma(s')\})^{\mathrm{it}}_s(v))$.

For each $M = \langle \Sigma, (V,E)\rangle \in |\mathcal{G}(Str^{\circ})|$ with $\Sigma = (S,D)$ and each morphism $\beta = \langle 1_\Sigma, \beta^{\mathrm{it}}\rangle : \langle \Sigma, X\rangle \to \mathcal{G}(U^{\circ})(M)$ define $(\beta)^{\natural^{\mathrm{it}}_\Sigma} = \langle 1_\Sigma, (\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_\Sigma}\rangle : \mathscr{C}^{\mathrm{it}}(\langle \Sigma, X\rangle) \to \mathcal{G}(U^{\circ})(M)$ inductively by

- $(\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(x) = \beta_s(x)$ for $x \in X_s$;
- $(\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(d(\vec{v})) = E_{\overline{s},s}(d)((\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_{\overline{s}}(\vec{v}))$;
- $(\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(v' \to \mathsf{iterate}(y' ; y = v_0 \,|\, v)) = it((\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_{s'}(v'), (\beta^{\mathrm{it}})^{\natural^{\mathrm{it}}_M}_s(v_0),$
  $\{(v_1, v_2) \mapsto ((\beta^{\mathrm{it}}\{y : s \mapsto v_2, y' : s' \mapsto v_1\})^{\natural^{\mathrm{it}}_M}_s(v)\})\})$,
  where $it(\varepsilon, v_a, f) = v_a$ and $it(v_i :: \ell, v_a, f) = it(\ell, f(v_i, v_a), f)$.

Term Charter. We obtain the *iteration term charter* $(\mathscr{C}^{\mathrm{it}}, \nu^{\mathrm{it}}, (-)^{\natural^{\mathrm{it}}})$ over the term charter domain $(\mathbb{S}^{\circ}, Val^{\leq}, Str^{\circ}, U^{\circ})$ when choosing the embedding natural transformation ν^{it} to consist out of inclusions. The evaluation of the iteration construct is completely handled by the structure over which a term is evaluated. As for primitives-closed order-sorted term charters we can construct the respective institutions.

A.4 All Instances

When accessing all instances of a type with an infinite number of inhabitants, an undefined value, which we denote by †, shall be the result. We cover this addition by lifting the order-sorted value domain morphisms to include also † in their co-domains.

Values. An *undef-lifting value domain morphism* $\omega^{\dagger} : V \to V'$ from an (S,D)-value domain V to an (S,D)-value domain V' over the order-sorted signature (S,D) is given by a family of mappings $\omega^{\dagger} = (\omega^{\dagger}_s : V_s \to (V'_s)_{\dagger})_{s \in |S|}$ where $(M)_{\dagger} = M \uplus \{\dagger\}$ is the *undef-lifting* of the set M extending M by the special *undefinedness symbol* †. The *composition* $\omega'^{\dagger} \circ \omega^{\dagger} : V \to V''$ of two undef-lifting value domain morphisms $\omega^{\dagger} : V \to V'$ and $\omega'^{\dagger} : V' \to V''$ between (S,D)-value domains is given by $\omega'^{\dagger} \circ \omega^{\dagger} = (\omega'^{\dagger}_s \circ \omega^{\dagger}_s : V_s \to (V''_s)_{\dagger})_{s \in |S|}$ with $(\omega'^{\dagger}_s \circ \omega^{\dagger}_s)(v) = \dagger$ if $\omega^{\dagger}_s(v) = \dagger$ and $(\omega'^{\dagger}_s \circ \omega^{\dagger}_s)(v) = \omega'^{\dagger}_s(\omega^{\dagger}_s(v))$ otherwise; i.e., composition is strict w.r.t. undefinedness. The *identity* undef-lifting value domain morphism $1^{\dagger}_V : V \to V$ between an (S,D)-value domain V is given by $1^{\dagger}_V = (1^{\dagger}_{V,s} : V_s \to (V_s)_{\dagger})_{s \in |S|}$ with $1^{\dagger}_{V,s}(v) = v$.

(S,D)-value domains and undef-lifting morphisms between (S,D)-value domains define a category which we denote by $Val^\dagger(S,D)$. The indexed category $Val^\dagger : (\mathbb{S}^{\leq})^{\mathrm{op}} \to \mathrm{Cat}$ maps each order-sorted signature (S,D) to $Val^\dagger(S,D)$ and each order-sorted signature morphism $(\gamma,\rho) : (S,D) \to (S',D')$ to the functor $Val^\dagger(\gamma,\rho) : Val^\dagger(S',D') \to Val^\dagger(S,D)$ with $Val^\dagger(\gamma,\rho)((V'_{s'})_{s'\in|S'|}) = (V'_{\gamma(s)})_{s\in|S|}$ and $Val^\dagger(\gamma,\rho)((\omega'^\dagger_{s'} : V'_{1,s'} \to (V'_{2,s'})_\dagger)_{s'\in|S|}) = (\omega'^\dagger_{\gamma(s)} : V'_{1,\gamma(s)} \to (V'_{2,\gamma(s)})_\dagger)_{s\in|S|}$.

For a primitives-closed $\Sigma = (S,D)$-structure (V,E) define $U^\dagger_\Sigma(V,E) = V$. For a structure morphism $\omega : (V,E) \to (V',E')$ between primitives-closed Σ-structures define $U^\dagger_\Sigma(\omega) = (\omega^\dagger_s : V_s \to (V'_s)_\dagger)_{s\in|S|}$ with $\omega^\dagger_s(v) = \omega(v)$ which is an undef-lifting value domain morphism from $U^\dagger_\Sigma(V,E)$ to $U^\dagger_\Sigma(V',E')$. This yields the indexed functor $U^\dagger : Str^\circ \dot{\to} Val^\dagger$.

Terms and Evaluation. The construction functor for *all-instances terms* $\mathscr{C}^{\mathrm{a}} : \mathcal{G}(Val^\dagger) \to \mathcal{G}(Val^\dagger)$ is defined as follows: For the objects, let $\mathscr{C}^{\mathrm{a}}(\langle (S,D),X\rangle) = \langle (S,D), V^{\mathrm{a}}_X\rangle$ such that inductively

- $x \in V^{\mathrm{a}}_{X,s}$ if $x \in X_s$;
- $d(\vec{v}) \in V^{\mathrm{a}}_{X,s'}$ for all $s' \geq_S s$ if $d \in \delta_D(\overline{s},s)$ and $\vec{v} \in V^{\mathrm{a}}_{X,\overline{s}}$;
- $s.\mathsf{allInstances}() \in V^{\mathrm{a}}_{X,s'}$ for all $s \in |S|$ and $s' \geq_S \mathsf{Set}(s)$;

For the morphisms, define $\mathscr{C}^{\mathrm{a}}(\langle(\gamma,\rho),\omega^\dagger\rangle) = \langle(\gamma,\rho),(\omega^\dagger)^{\mathrm{a}}\rangle : \mathscr{C}^{\mathrm{a}}(\langle(S,D),X\rangle) \to \mathscr{C}^{\mathrm{a}}(\langle(S',D'),X'\rangle)$ such that, by simultaneous induction,

- $(\omega^\dagger)^{\mathrm{a}}_s(x) = \omega^\dagger_s(x)$ for $x \in X_s$;
- $(\omega^\dagger)^{\mathrm{a}}_{s'}(d(\vec{v})) = \begin{cases} \rho(d)((\omega^\dagger)^{\mathrm{a}}_{\overline{s}}(\vec{v})) & \text{if } \omega^\dagger_{\overline{s}_i}(\vec{v}_i) \neq \dagger \text{ for all } 1 \leq i \leq |\vec{v}| \\ \dagger & \text{otherwise} \end{cases}$;
- $(\omega^\dagger)^{\mathrm{a}}_{s'}(s.\mathsf{allInstances}()) = \gamma(s).\mathsf{allInstances}()$.

For each $M = \langle\Sigma,(V,E)\rangle$ in $|\mathcal{G}(Str^\circ)|$ with $\Sigma = (S,D)$ and each $\beta = \langle 1_\Sigma,\beta^{\mathrm{a}}\rangle : \langle\Sigma,X\rangle \to \mathcal{G}(U^\dagger)(M)$ define $(\beta)^{\natural^{\mathrm{a}}_M} = \langle 1_\Sigma,(\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}\rangle : \mathscr{C}^{\mathrm{a}}(\langle\Sigma,X\rangle) \to \mathcal{G}(U^\dagger)(M)$ inductively by

- $(\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_s(x) = \beta^{\mathrm{a}}_s(x)$ for $x \in X_s$;
- $(\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_{s'}(d(\vec{v})) = \begin{cases} E_{\overline{s},s}(d)((\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_{\overline{s}}(\vec{v})) & \text{if } (\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_{\overline{s}_i}(\vec{v}_i) \neq \dagger \text{ for all } 1 \leq i \leq |\vec{v}| \\ \dagger & \text{otherwise} \end{cases}$;
- $(\beta^{\mathrm{a}})^{\natural^{\mathrm{a}}_M}_{s'}(s.\mathsf{allInstances}()) = \begin{cases} V_s & \text{if } |V_s| < \infty \\ \dagger & \text{otherwise} \end{cases}$.

Term Charter. Define the embedding natural transformation $\nu^{\mathrm{a}} : 1_{\mathcal{G}(Val^\dagger)} \dot{\to} \mathscr{C}^{\mathrm{a}}$ again as inclusions, though now as undef-lifting value domain morphisms, i.e., $\nu^{\mathrm{a}}_{\langle(S,D),X\rangle} = \langle 1_{(S,D)},(\iota^{\mathrm{a}}_{\langle(S,D),X\rangle,s} : X_s \to (V^{\mathrm{a}}_{X,s})_\dagger)_{s\in|S|}\rangle$. This yields the *all-instances term charter* $(\mathscr{C}^{\mathrm{a}},\nu^{\mathrm{a}},(-)^{\natural^{\mathrm{a}}})$ over the term charter domain $(\mathbb{S}^\circ, Val^\dagger, Str^\circ, U^\dagger)$, though checking the term charter conditions (C), (K), and (E) becomes a little bit more tedious because of case distinctions.

A.5 Undefinedness

Finally, let us consider OCL's handling of undefinedness. We also add an if-then-else clause as another non-strict function besides the test on undefinedness and the three-valued Boolean connectives.

Terms and Evaluation. The construction functor for *undefinedness terms* $\mathscr{C}^{\mathsf{u}} : \mathcal{G}(Val^{\dagger}) \to \mathcal{G}(Val^{\dagger})$ is defined as follows: For the objects, set $\mathscr{C}^{\mathsf{u}}(\langle (S,D), X\rangle) = \langle (S,D), V^{\mathsf{u}}_X\rangle$ such that inductively

- $x \in V^{\mathsf{u}}_{X,s}$ if $x \in X_s$;
- $d(\vec{v}) \in V^{\mathsf{u}}_{X,s'}$ for all $s' \geq_S s$ if $d \in \delta_D(\overline{s}, s)$ and $\vec{v} \in V^{\mathsf{u}}_{X,\overline{s}}$;
- $\mathsf{undef} \in V^{\mathsf{u}}_{X,s}$ for all $s \in |S|$;
- $v.\mathsf{isUndef}() \in V^{\mathsf{u}}_{X,s'}$ for all $s' \geq_S \mathsf{Bool}$ if $v \in V^{\mathsf{u}}_{X,s}$ for $s \in |S|$;
- v_1 and $v_2 \in V^{\mathsf{u}}_{X,s}$ and v_1 or $v_2 \in V^{\mathsf{u}}_{X,s}$ for all $v_1, v_2 \in V^{\mathsf{u}}_{X,\mathsf{Bool}}$ and $s \geq_S \mathsf{Bool}$;
- if v then v_1 else v_2 endif $\in V^{\mathsf{u}}_{X,s'}$ for all $v \in V^{\mathsf{u}}_{X,\mathsf{Bool}}$ and $v_1, v_2 \in V^{\mathsf{u}}_{X,s}$ with $s' \geq_S s$.

For the morphisms, define $\mathscr{C}^{\mathsf{u}}(\langle (\gamma,\rho), \omega^{\dagger}\rangle) = \langle (\gamma,\rho), (\omega^{\dagger})^{\mathsf{u}}\rangle : \mathscr{C}^{\mathsf{u}}(\langle (S,D), X\rangle) \to \mathscr{C}^{\mathsf{u}}(\langle (S',D'), X'\rangle)$ such that, by simultaneous induction,

- $(\omega^{\dagger})^{\mathsf{u}}_s(x) = \begin{cases} \mathsf{undef} & \text{if } \omega^{\dagger}_s(x) = \dagger \\ \omega^{\dagger}_s(x) & \text{otherwise} \end{cases}$ for $x \in X_s$;
- $(\omega^{\dagger})^{\mathsf{u}}_{s'}(d(\vec{v})) = \rho(d)((\omega^{\dagger})^{\mathsf{u}}_{\overline{s}}(\vec{v}))$ for $d \in \delta_D(\overline{s}, s)$;
- $(\omega^{\dagger})^{\mathsf{u}}_s(\mathsf{undef}) = \mathsf{undef}$;
- $(\omega^{\dagger})^{\mathsf{u}}_{s'}(v.\mathsf{isUndef}()) = (\omega^{\dagger})^{\mathsf{u}}_s(v).\mathsf{isUndef}()$;
- $(\omega^{\dagger})^{\mathsf{u}}_s(v_1\ bop\ v_2) = (\omega^{\dagger})^{\mathsf{u}}_{\mathsf{Bool}}(v_1)\ bop\ (\omega^{\dagger})^{\mathsf{u}}_{\mathsf{Bool}}(v_2)$ for $bop \in \{\mathsf{and}, \mathsf{or}\}$;
- $(\omega^{\dagger})^{\mathsf{u}}_{s'}(\mathsf{if}\ v\ \mathsf{then}\ v_1\ \mathsf{else}\ v_2\ \mathsf{endif}) = \mathsf{if}\ (\omega^{\dagger})^{\mathsf{u}}_{\mathsf{Bool}}(v)\ \mathsf{then}\ (\omega^{\dagger})^{\mathsf{u}}_s(v_1)\ \mathsf{else}\ (\omega^{\dagger})^{\mathsf{u}}_s(v_2)\ \mathsf{endif}$.

(Although $(\omega^{\dagger})^{\mathsf{u}}_s(x) = \omega^{\dagger}_s(x)$ together with a strict extension for $d(\vec{v})$ could have been defined, special measures for if v then v_1 else v_2 endif would have to be taken.)

For each $M = \langle \Sigma, (V,E)\rangle$ in $|\mathcal{G}(Str^{\circ})|$ with $\Sigma = (S,D)$ and each $\beta = \langle 1_{\Sigma}, \beta^{\mathsf{u}}\rangle : \langle \Sigma, X\rangle \to \mathcal{G}(U^{\dagger})(M)$ define $(\beta)^{\natural^{\mathsf{u}}_M} = \langle 1_{\Sigma}, (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}\rangle : \mathscr{C}^{\mathsf{u}}(\langle \Sigma, X\rangle) \to \mathcal{G}(U^{\dagger})(M)$ inductively by

- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(x) = \beta^{\mathsf{u}}_s(x)$ for $x \in X_s$;
- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{s'}(d(\vec{v})) = \begin{cases} E_{\overline{s},s}(d)((\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\overline{s}}(\vec{v})) & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\overline{s}_i}(\vec{v}_i) \neq \dagger \text{ for all } 1 \leq i \leq |\vec{v}| \\ \dagger & \text{otherwise} \end{cases}$;
- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(\mathsf{undef}) = \dagger$;
- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{s'}(v.\mathsf{isUndef}()) = \begin{cases} tt & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v) = \dagger \\ ff & \text{otherwise} \end{cases}$;
- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v_1\ \mathsf{and}\ v_2) = \begin{cases} tt & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_1) = tt \text{ and } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_2) = tt \\ ff & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_1) = ff \text{ or } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_2) = ff \\ \dagger & \text{otherwise} \end{cases}$;
- $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v_1\ \mathsf{or}\ v_2) = \begin{cases} tt & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_1) = tt \text{ or } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_2) = tt \\ ff & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_1) = ff \text{ and } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_{\mathsf{Bool}}(v_2) = ff \\ \dagger & \text{otherwise} \end{cases}$;

– $(\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(\mathsf{if}\ v\ \mathsf{then}\ v_1\ \mathsf{else}\ v_2\ \mathsf{endif}) = \begin{cases} (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v_1) & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v) = tt \\ (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v_2) & \text{if } (\beta^{\mathsf{u}})^{\natural^{\mathsf{u}}_M}_s(v) = ff \\ \dagger & \text{otherwise} \end{cases}$.

Term Charter. The *undefinedness term charter* $(\mathscr{C}^{\mathsf{u}}, \nu^{\mathsf{u}}, (-)^{\natural^{\mathsf{u}}})$ over the term charter domain $(\mathbb{S}^\circ, Val^\dagger, Str^\circ, U^\dagger)$ uses the analogous embedding natural transformation $\nu^{\mathsf{u}} : 1_{\mathcal{G}(Val^\dagger)} \dot{\to} \mathscr{C}^{\mathsf{u}}$ as the all-instances term charter. Checking the term charter conditions (C), (K), and (E) now involves even more case distinctions. In contrast to the all-instances case, the undef-lifting value domain morphism constructed by $\mathscr{C}^{\mathsf{u}}$ not simply is a strict extension to terms, but treats † specially in order to avoid problems with the if-then-else clause.