Externalized and Internalized Notions of Behavioral Refinement *

Michel Bidoit [1] Rolf Hennicker [2]

[1] Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan, France
[2] Institut für Informatik, Ludwig-Maximilians-Universität München, Germany

Abstract. Many different behavioral refinement notions for algebraic specifications have been proposed in the literature but the relationship between the various concepts is still unclear. In this paper we provide a classification and a comparative study of behavioral refinements according to two directions, the *externalized* approach which uses an explicit behavioral abstraction operator that is applied to the specification to be implemented, and the *internalized* approach which uses a built-in behavioral semantics of specifications. We show that both concepts are equivalent under suitable conditions. The formal basis of our study is provided by the COL institution (constructor-based observational logic). Hence, as a side-effect of our study on internalized behavioral refinements, we introduce also a novel concept of behavioral refinement for COL-specifications.

1 Introduction

The investigation of behavioral refinement notions is motivated by the fact that, in general, an implementation does not need to satisfy literally the properties of an abstract specification but it can nevertheless be considered as correct if this implementation respects the observable consequences of the specification to be implemented. In the framework of algebraic specifications this idea has been taken into account in many approaches in the literature proposing behavioral (or observational) refinement (or implementation) concepts; see e.g. [9, 17, 19, 12, 13, 4] and, for an overview, [16, 8]. However, due to the various different formalizations, there is still no clear picture of the relationships between the various approaches.

In this paper we propose a classification based on two principal directions that can be identified when we analyze behavioral refinement concepts. The first direction, in the following called the *externalized view*, uses an explicit behavioral abstraction operator to relax the (semantics of the) specification to be implemented. The general idea is then that the models of an implementing specification not necessarily have to lie in the model class of the specification to be implemented but it is sufficient if they lie in its "abstracted" model class

* This work is partially supported by the German BMBF-project GLOWA-Danube.

(see e.g. [17, 19, 16, 4]). Of course, there is again a variety of proposed behavioral abstraction operators which are either based on observational equivalences between algebras (see, e.g., [17, 16]) or on observational equalities between the elements of algebras (see, e.g., [4]). Since in many cases both approaches can be expressed by each other (see [7]), we will restrict here to behavioral abstraction operators that are based on observational equalities between elements. As a concrete formalism we use the notion of observational equality defined in [5] which is based on distinguished sets of constructor operations (determining the relevant values from the user's point of view) and observer operations (determining the indistinguishability of elements). As a first result, we show in Section 3 (Theorem 1) that behavioral refinement relations based on the externalized view can be characterized by standard, non-behavioral refinements if we use a quotient construction as an implementation constructor.

Then, in Section 4, we consider the second direction to behavioral refinement, in the following called the *internalized view*. Here the idea is to use a built-in behavioral semantics that is used both for the specification to be implemented and for the implementing specification. A built-in behavioral semantics is most appropriately obtained by the use of a behavioral institution that provides a logical system focusing on the behavioral aspects of system specifications (as with hidden algebra [10] or the constructor-based observational logic COL [5]). A behavioral refinement concept based on hidden algebra is studied in [13], a behavioral refinement concept for COL-specifications is introduced in Section 4. This refinement concept is based on the notion of a COL-implementation constructor which can be applied to the models of the implementing COL-specification $\mathrm{SPI}_{\mathrm{COL}}$ to produce models of the COL-specification $\mathrm{SP}_{\mathrm{COL}}$ to be implemented. A crucial property of COL-implementation constructors is that they have to be compatible with behavioral isomorphisms. We show that under mild assumptions reduct functors along (standard) signature morphisms are indeed COL-implementation constructors (Lemma 1) and we discuss the need for such constructions (in contrast to reduct functors along COL-signature morphisms which are appropriate for encapsulation of specifications but not adequate for refinement).

In Section 5 we discuss the relationships between the externalized and the internalized views on behavioral refinements. We show that the behavioral compatibility assumption of COL-implementation constructors is closely related to the notion of stability (introduced by Schoett [20]) which requires that implementation constructors preserve observational equivalences between algebras. Indeed, considering the externalized view, stability is the crucial criterion to obtain composability of behavioral refinement steps (see [19]), also called *vertical composition*. For the internalized view vertical composition of behavioral refinements is guaranteed by definition, according to the built-in behavioral semantics of the implementing specification. As the central result of this paper we show in Theorem 2 that, under suitable assumptions, externalized and internalized notions of behavioral refinement can be expressed by each other. As pointed out in Section 6, this leads to a useful proof rule for internalized behavioral refinements.

2 Basic Concepts

2.1 Algebraic Preliminaries and Structured Specifications

We assume that the reader is familiar with the basic notions of algebraic specifications (see, e.g., [22, 1]), like the notions of (many-sorted) *signature* $\Sigma = (S, \mathrm{OP})$ (where S is a set of *sorts* and OP is a set of *operation symbols* $op : s_1, \ldots, s_n \to s$), *signature morphism* $\sigma : \Sigma \to \Sigma'$, *(total)* Σ*-algebra* $A = ((A_s)_{s \in S}, (op^A)_{op \in \mathrm{OP}})$, The class of all Σ-algebras is denoted by $\mathrm{Alg}(\Sigma)$. Together with Σ-morphisms this class forms a category which, for simplicity, is also denoted by $\mathrm{Alg}(\Sigma)$. For any signature morphism $\sigma : \Sigma \to \Sigma'$, the *reduct functor* $\_|_\sigma : \mathrm{Alg}(\Sigma') \to \mathrm{Alg}(\Sigma)$ is defined as usual.

The notion of an institution was introduced by Goguen and Burstall [11] to formalize the general concept of a logical system from a model-theoretic point of view; see [21] for an overview on the basic definitions and the theory of institutions. Any institution provides a suitable framework for defining a set of specification-building operators which are independent from the concrete form of the institution. We will use the following four fundamental operators introduced in [18] for constructing structured specifications over an institution I. The semantics of a specification SP is always determined by its signature, denoted by $\mathcal{S}ig[\mathrm{SP}]$, and by its class of models, denoted by $\mathcal{M}od[\mathrm{SP}]$.

***presentation*:** Any pair $\langle \Sigma, \Phi \rangle$ consisting of a signature Σ and of a set Φ of Σ-sentences is a specification with semantics:
$\mathcal{S}ig[\langle \Sigma, \Phi \rangle] \stackrel{\mathrm{def}}{=} \Sigma$
$\mathcal{M}od[\langle \Sigma, \Phi \rangle] \stackrel{\mathrm{def}}{=} \{M \in \mathrm{Mod}(\Sigma) \mid M \models_\Sigma \Phi\}$

***union*:** For any two specifications SP_1 and SP_2 with the same signature $\mathcal{S}ig[\mathrm{SP}_1] = \mathcal{S}ig[\mathrm{SP}_2] = \Sigma$, the expression $\mathrm{SP}_1 \cup \mathrm{SP}_2$ is a specification with semantics:
$\mathcal{S}ig[\mathrm{SP}_1 \cup \mathrm{SP}_2] \stackrel{\mathrm{def}}{=} \Sigma$
$\mathcal{M}od[\mathrm{SP}_1 \cup \mathrm{SP}_2] \stackrel{\mathrm{def}}{=} \mathcal{M}od[\mathrm{SP}_1] \cap \mathcal{M}od[\mathrm{SP}_2]$

***translation*:** For any specification SP and signature morphism $\sigma : \mathcal{S}ig[\mathrm{SP}] \to \Sigma$, the expression **translate** SP **by** σ is a specification with semantics:
$\mathcal{S}ig[\textbf{translate SP by } \sigma] \stackrel{\mathrm{def}}{=} \Sigma$
$\mathcal{M}od[\textbf{translate SP by } \sigma] \stackrel{\mathrm{def}}{=} \{M \in \mathrm{Mod}(\Sigma) \mid M|_\sigma \in \mathcal{M}od[\mathrm{SP}]\}$

***hiding*:** For any specification SP and signature morphism $\sigma : \Sigma \to \mathcal{S}ig[\mathrm{SP}]$, the expression **derive from** SP **by** σ is a specification with semantics:
$\mathcal{S}ig[\textbf{derive from SP by } \sigma] \stackrel{\mathrm{def}}{=} \Sigma$
$\mathcal{M}od[\textbf{derive from SP by } \sigma] \stackrel{\mathrm{def}}{=} \mathrm{Iso}_\Sigma(\{M|_\sigma \mid M \in \mathcal{M}od[\mathrm{SP}]\})$,
where $\mathrm{Iso}_\Sigma(\_)$ denotes the closure under Σ-isomorphisms in $\mathrm{Mod}(\Sigma)$.

2.2 Observability Concepts

In this section we recall the underlying observability notions that will be used hereafter to formalize behavioral refinements (see [5] for more details). Note, however, that the forthcoming study of behavioral refinement notions is in principle independent of the chosen formal basis.

To capture the behavioral aspects of system specifications we consider distinguished sets of constructor and observer operations. Intuitively, the constructor operations determine those elements which are of interest from the user's point of view while the observers determine a set of observable experiments that a user can perform to examine hidden states. Thus we can abstract from junk elements and also from concrete state representations whereby two states are considered to be "observationally equal" if they cannot be distinguished by observable experiments.

Formally, a *constructor operation* is an operation symbol $cons : s_1, \ldots, s_n \rightarrow s$ with $n \geq 0$. The result sort s of *cons* is called a *constrained sort*. An *observer operation* is a pair (obs, i) where *obs* is an operation symbol $obs : s_1, \ldots, s_n \rightarrow s$ with $n \geq 1$ and $1 \leq i \leq n$. The distinguished argument sort s_i of *obs* is called a *state sort* (or *hidden sort*). If $obs : s_1 \rightarrow s$ is a unary observer we simply write *obs* instead of $(obs, 1)$.

If we consider a standard algebraic signature $\Sigma = (S, \mathrm{OP})$ together with a distinguished set $\mathrm{OP}_{\mathrm{Cons}}$ of constructor operations and a distinguished set $\mathrm{OP}_{\mathrm{Obs}}$ of observer operations we obtain a so-called COL-*signature* $\Sigma_{\mathrm{COL}} = (\Sigma, \mathrm{OP}_{\mathrm{Cons}}, \mathrm{OP}_{\mathrm{Obs}})$ with underlying (standard) signature Σ.[1] The set $S_{\mathrm{Cons}} \subseteq S$ of *constrained sorts* (w.r.t. $\mathrm{OP}_{\mathrm{Cons}}$) consists of all sorts s such that there exists at least one constructor in $\mathrm{OP}_{\mathrm{Cons}}$ with range s. The set $S_{\mathrm{Loose}} \subseteq S$ of *loose sorts* consists of all non-constrained sorts, i.e. $S_{\mathrm{Loose}} = S \setminus S_{\mathrm{Cons}}$. The set $S_{\mathrm{State}} \subseteq S$ of *state sorts* (or *hidden sorts*, w.r.t. $\mathrm{OP}_{\mathrm{Obs}}$) consists of all sorts s_i such that there exists at least one observer (obs, i) in $\mathrm{OP}_{\mathrm{Obs}}$, $obs : s_1, \ldots, s_i, \ldots, s_n \rightarrow s$. The set $S_{\mathrm{Obs}} \subseteq S$ of *observable sorts* consists of all sorts which are not a state sort, i.e. $S_{\mathrm{Obs}} = S \setminus S_{\mathrm{State}}$.

The set $\mathrm{OP}_{\mathrm{Cons}}$ of constructor operations (of a COL-signature Σ_{COL}) determines a set of *constructor terms*. A constructor term is a term t of a constrained sort $s \in S_{\mathrm{Cons}}$ which is built only from constructor operations of $\mathrm{OP}_{\mathrm{Cons}}$ and from variables of loose sorts. In particular, if all sorts are constrained, i.e., $S_{\mathrm{Cons}} = S$, the constructor terms are exactly the $(S, \mathrm{OP}_{\mathrm{Cons}})$-ground terms which are built by the constructor symbols. The set of constructor terms determines, for any Σ-algebra A, a family of subsets of the carrier sets of A, called the *generated part* and denoted by $\mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)$, which consists of those elements that can be constructed by the interpretations of the given constructors (starting from constants and from arbitrary elements of loose sorts, if any). The Σ_{COL}-generated part represents those elements which are of interest from the

[1] The terminology "COL-signature" stems from the constructor-based observational logic institution COL. Our study is however independent from the COL institution as long as we do not consider the internalized view of behavioral refinements studied in Section 4.

user's point of view according to the given constructor operations. A Σ-algebra A is *reachable* (w.r.t. Σ_{COL}) if its carrier sets coincide with the carrier sets of its Σ_{COL}-generated part.

The set $\mathrm{OP}_{\mathrm{Obs}}$ of observer operations determines a set of *observable contexts* which represent the observable experiments that a user can perform. An observable context is a term c of observable sort $s' \in S_{\mathrm{Obs}}$ which is built only from observer operations of $\mathrm{OP}_{\mathrm{Obs}}$ and which contains a distinguished variable z_s of some hidden sort $s \in S_{\mathrm{State}}$. s is called the *application sort* and s' is called the *observable result sort* of c. The set of observable contexts determines, for any Σ-algebra A, an indistinguishability relation, called *observational equality* and denoted by $\approx_{\Sigma_{\mathrm{COL}},A}$. For any two elements $a, b \in A$, $a \approx_{\Sigma_{\mathrm{COL}},A} b$ holds if either $a = b$ and a, b are observable (i.e. belong to a carrier set of observable sort $s \in S_{\mathrm{Obs}}$) or if a and b cannot be distinguished by the application of observable contexts. A Σ-algebra A is *fully abstract* if the observational Σ_{COL}-equality coincides with the set-theoretic equality.

The constructor and the observer operations induce certain constraints on Σ-algebras. First, since the constructor operations determine the values of interest, we require that the non-constructor operations should (up to observational equality) respect the constructor-generated part of an algebra, i.e. by the application of non-constructor operations one should at most be able to obtain elements which are observationally equal to some element of the constructor-generated part. Technically this means that for a given Σ-algebra A we first consider the smallest Σ-subalgebra $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A) \rangle_\Sigma$ of A containing the Σ_{COL}-generated part because this subalgebra represents the only elements a user can compute (over the loose carrier sets) by invoking operations of Σ. Then we require that each element of $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A) \rangle_\Sigma$ is observationally equal to some element of the Σ_{COL}-generated part $\mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)$ of A. This condition is called *reachability constraint*.

Secondly, since the declaration of observer operations determines a particular observational equality on any Σ-algebra A, the (interpretations of the) non-observer operations should respect this observational equality, i.e. a non-observer operation should not contribute to distinguish non-observable elements. To ensure this we require that the observational equality is a Σ-congruence on the subalgebra $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A) \rangle_\Sigma$. (Note that it is sufficient to consider $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A) \rangle_\Sigma$ instead of A because computations performed by a user can only lead to elements in the Σ-subalgebra $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A) \rangle_\Sigma$.) This condition is called *observability constraint*.

A Σ-algebra A which satisfies both the reachability and the observability constraints induced by a COL-signature $\Sigma_{\mathrm{COL}} = (\Sigma, \mathrm{OP}_{\mathrm{Cons}}, \mathrm{OP}_{\mathrm{Obs}})$ is called Σ_{COL}-*algebra* (or simply COL-*algebra*). Note that any Σ-algebra A which is reachable and fully abstract w.r.t. Σ_{COL} is a Σ_{COL}-algebra. The class of all Σ_{COL}-algebras is denoted by $\mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}})$.

The satisfaction of the reachability and observability constraints allows us to construct for each Σ_{COL}-algebra A its *black box view* which is a reachable and fully abstract algebra representing the behavior of A from the user's point

of view. The black box view is constructed in two steps. First, we *restrict* to the Σ_{COL}-generated subalgebra $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)\rangle_{\Sigma}$ of A thus forgetting junk values. Then, we *identify* all elements of $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)\rangle_{\Sigma}$ which are observationally equal. Hence the black box view of a Σ_{COL}-algebra A is given by the quotient algebra of $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)\rangle_{\Sigma}$ w.r.t. $\approx_{\Sigma_{\mathrm{COL}},A}$ which, for simplicity, will be denoted by $A/\approx_{\Sigma_{\mathrm{COL}},A}$. Two Σ_{COL}-algebras A and B are *observationally equivalent* (w.r.t. Σ_{COL}), denoted by $A \equiv_{\Sigma_{\mathrm{COL}}} B$, if their black box views $A/\approx_{\Sigma_{\mathrm{COL}},A}$ and $B/\approx_{\Sigma_{\mathrm{COL}},B}$ are isomorphic Σ-algebras.

The observability notions defined above provide a generalization of the approach in [7] which is based on partial observational equalities $\approx_{Obs,In,A}$. The difference here is the declaration of the constructor and observer operations which provide much more flexibility than declaring just observable sorts *Obs* and input sorts *In* as done in [7]. In fact, any standard signature $\Sigma = (S, \mathrm{OP})$ together with distinguished sets $In \subseteq S$ of input sorts and $Obs \subseteq S$ of observable sorts induces a COL-signature $\Sigma_{\mathrm{COL}}^{\mathrm{In,Obs}} = (\Sigma, \mathrm{OP}_{\mathrm{Cons}}, \mathrm{OP}_{\mathrm{Obs}})$ where $\mathrm{OP}_{\mathrm{Cons}}$ consists of *all* operation symbols $cons \in \mathrm{OP}$ with range $s \in S \setminus In$ and $\mathrm{OP}_{\mathrm{Obs}}$ consists of all pairs (obs, i) with $obs \in \mathrm{OP}$, $obs : s_1, \ldots, s_i, \ldots, s_n \to s$ and $s_i \in S \setminus Obs$. Then, for any Σ-algebra A, the partial observational equality $\approx_{Obs,In,A}$ coincides (on $\langle \mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)\rangle_{\Sigma}$) with $\approx_{\Sigma_{\mathrm{COL}},A}$. In particular, in this case each Σ-algebra is also a COL-algebra. Hence the results on behavioral refinements developed in the following sections are also valid for all observability notions based on fixed sets of observable sorts (and input sorts) which are frequently found in the literature, see, e.g., [15, 17].

3 Behavioral Refinement: The Externalized View

In this section we consider the institution FOLEq of many-sorted first-order logic with equality (as detailed, e.g., in [3]) and we consider structured specifications over FOLEq built by the specification building operations defined in Section 2.1. A simple refinement relation between two specifications SP (the abstract specification to be implemented) and SPI (the implementing specification) can be defined by requiring that both specifications have the same signature and that the model class of the implementing specification SPI is included in the model class of SP, see, e.g., [22]. To take into account that an implementation usually involves some construction steps the notion of constructor implementation has been introduced in [19] (and similarly in other implementation concepts; see [16, 8] for an overview). According to [19] an implementation constructor is a function which maps algebras over the signature of the implementing specification to algebras over the signature of the abstract specification. Since an implementation construction must not necessarily be defined on all algebras but only on the models of the implementing specification we allow partial functions as implementation constructors. (An example of a partial implementation constructor is the formation of observational quotients used below.) On the other hand, we assume that implementation constructions are performed in a uniform way, i.e. preserve isomorphisms.

Definition 1 (Implementation constructor). *Let Σ, ΣI be two signatures. An* implementation constructor *from ΣI to Σ (also simply called a* constructor*) is a partial function $\kappa : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ which is iso-preserving, i.e. for all $AI, BI \in \mathrm{Alg}(\Sigma I)$,*

if AI is ΣI-isomorphic to BI and $\kappa(AI)$ is defined
then $\kappa(BI)$ is defined and $\kappa(AI)$ is Σ-isomorphic to $\kappa(BI)$.

The definition domain of κ is denoted by $Dom(\kappa)$.

An example of an implementation constructor is, for a given signature morphism $\sigma : \Sigma \to \Sigma I$ which renames abstract sorts and operations into those offered by the implementation, the reduct functor $\_\_|_\sigma : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ (see also [19]). Note that in FOLEq this constructor can also be expressed by the *derive* specification-building primitive.

Definition 2 (Refinement). *Let* SP*,* SPI *be two specifications with signatures Σ, ΣI resp. and let κ be a constructor from ΣI to Σ.* SPI *is a* refinement *of* SP w.r.t. κ*, denoted by* SP $\rightsquigarrow^{\kappa}$ SPI*, if*

$$\mathcal{M}od[\mathrm{SPI}] \subseteq Dom(\kappa) \text{ and } \kappa(\mathcal{M}od[\mathrm{SPI}]) \subseteq \mathcal{M}od[\mathrm{SP}].$$

Many examples show that the above refinement definition is too restrictive since an implementation does not need to satisfy literally all requirements of an abstract specification but can nevertheless be considered as correct if the implementation respects the observable properties of the specification to be implemented. This fact has inspired a lot of work on adequate notions of behavioral refinement relations. A popular idea is to relax the model class of the specification SP to be implemented by some behavioral abstraction operation, see, e.g., [17, 19, 16, 4]. We call this direction the *externalized view* of behavioral refinement because, only for the purpose of refinement, a behavioral abstraction operation is applied on top of the given (standard) model class of SP. In contrast to that idea, other approaches use a built-in behavioral semantics which is used for both specifications, the specification to be implemented *and* the implementing specification, see [13]. We call this direction the *internalized view* of behavioral refinement which will be more closely considered in the next section. In this section we focus on the externalized view using as a behavioral abstraction operation the following behavior operator which constructs for a given class C of Σ-algebras the class of all algebras whose black box view belongs to C. The behavior operator is defined according to distinguished sets of constructor operations and observer operations, i.e. w.r.t. a COL-signature.

Definition 3 (Behavior operator). *Let $\Sigma_{\mathrm{COL}} = (\Sigma, \mathrm{OP}_{\mathrm{Cons}}, \mathrm{OP}_{\mathrm{Obs}})$ be a* COL*-signature. For any class C of Σ-algebras,*

$$\mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C) \stackrel{\mathrm{def}}{=} \{A \in \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}}) \mid A/{\approx_{\Sigma_{\mathrm{COL}},A}} \in C\}.$$

A class C of Σ-algebras is called *behaviorally closed* w.r.t. a COL-signature Σ_{COL} if $C \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$ or, equivalently, if any Σ-algebra $A \in C$ is a COL-algebra and its black box view $A/{\approx_{\Sigma_{\mathrm{COL}},A}}$ belongs also to C. A specification SP is behaviorally closed if its model class $\mathcal{M}od[\mathrm{SP}]$ is behaviorally closed.

When considering the externalized view of behavioral refinement the idea is, of course, to apply the behavior operator to the model class of the specification to be implemented. This leads to the following notion of behavioral refinement.

Definition 4 (Behavioral refinement: the externalized view). *Let* SP, SPI *be two specifications with signatures Σ, ΣI resp., let Σ_{COL} be a* COL*-signature of the form $(\Sigma, \mathrm{OP}_{\mathrm{Cons}}, \mathrm{OP}_{\mathrm{Obs}})$ and let $\kappa : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ be a constructor.* SPI *is a* behavioral refinement *of* SP w.r.t. Σ_{COL} and κ*, denoted by* SP $\overset{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa}$ SPI*, if*

$$\mathcal{M}od[\mathrm{SPI}] \subseteq Dom(\kappa) \text{ and } \kappa(\mathcal{M}od[\mathrm{SPI}]) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}]).$$

The given behavioral refinement notion is essentially based on the use of the observational equality of elements induced by a COL-signature. Other approaches in the literature, which follow the externalized view, use for behavioral abstraction not an indistinguishability relation between elements but an abstraction equivalence between algebras, see, e.g., [17, 15]. According to the results in [7, 4] there is, however, no difference between both approaches if the abstraction equivalence is factorizable (see [7]) and if the specification to be implemented is behaviorally closed.

Example 1. The following specification SET specifies properties of sets of natural numbers.

spec SET =
sorts *bool*, *nat*, *set*
ops *true*, *false* : *bool*;
0 : *nat*; *succ* : *nat* → *nat*; *plus* : *nat* × *nat* → *nat*;
empty : *set*; *add* : *nat* × *set* → *set*;
isin : *nat* × *set* → *bool*;
axioms
$\forall x, y : nat;\ s : set$
%% standard axioms for booleans and natural numbers, plus
- $isin(x, empty) = false$
- $isin(x, add(x, s)) = true$
- $x \neq y \Rightarrow isin(x, add(y, s)) = isin(x, s)$
- $add(x, add(x, s)) = add(x, s)$ (1)
- $add(x, add(y, s)) = add(y, add(x, s))$ (2)

end

For the implementation of sets we first abstract from the SET specification by using as an observer operation the membership test *isin* to observe sets. More precisely, we consider the COL-signature $\Sigma\mathrm{SET}_{\mathrm{COL}} = (\mathcal{S}ig[\mathrm{SET}], \emptyset, \{(isin, 2)\})$. For the concrete implementation we use the specification LIST shown below and a signature morphism $\sigma_{\mathrm{SET}as\mathrm{LIST}} : \mathcal{S}ig[\mathrm{SET}] \to \mathcal{S}ig[\mathrm{LIST}]$ such that $\sigma_{\mathrm{SET}as\mathrm{LIST}}(set) = list, \sigma_{\mathrm{SET}as\mathrm{LIST}}(add) = cons$ and $\sigma_{\mathrm{SET}as\mathrm{LIST}}(x) = x$ otherwise. Hence the implementation constructor κ is the reduct functor $\_|_{\sigma_{\mathrm{SET}as\mathrm{LIST}}} : \mathrm{Alg}(\mathcal{S}ig[\mathrm{LIST}]) \to \mathrm{Alg}(\mathcal{S}ig[\mathrm{SET}])$.

Thus we obtain the refinement relation $\textsc{Set}\ \overset{\Sigma\textsc{Set}_{\textsc{COL}}}{\rightsquigarrow^{\kappa}}\ \textsc{List}$.[2]

spec LIST =
sorts *bool*, *nat*, *list*
ops *true*, *false* : *bool*;
0 : *nat*; *succ* : *nat* → *nat*; *plus* : *nat* × *nat* → *nat*;
empty : *list*; *cons* : *nat* × *list* → *list*;
head : *list* → *nat*; *tail* : *list* → *list*;
isin : *nat* × *list* → *bool*;
axioms
$\forall x, y : nat;\ l : list$
%% standard axioms for booleans and natural numbers, plus
- $head(cons(x, l)) = x$
- $tail(cons(x, l)) = l$
- $isin(x, empty) = false$
- $isin(x, cons(x, l)) = true$
- $x \neq y \Rightarrow isin(x, cons(y, l)) = isin(x, l)$

end

Let us still point out that inspired by the results in [3] we can characterize externalized behavioral refinements by standard refinements in the sense of Definition 2 if we use behavioral quotient constructors which are induced by the black box views of COL-algebras.

Definition 5 (Behavioral quotient constructor). *Let $\Sigma_{\textsc{COL}}$ be a* COL*-signature with underlying signature Σ. The* behavioral quotient constructor (w.r.t. $\Sigma_{\textsc{COL}}$) *is given by* $\_\_/{\approx_{\Sigma_{\textsc{COL}}}} : \mathrm{Alg}(\Sigma) \to \mathrm{Alg}(\Sigma)$, *where*
$\_\_/{\approx_{\Sigma_{\textsc{COL}}}}(A) \overset{\text{def}}{=} A/{\approx_{\Sigma_{\textsc{COL}},A}}$ *if A is a $\Sigma_{\textsc{COL}}$-algebra,*
$\_\_/{\approx_{\Sigma_{\textsc{COL}}}}(A)$ *is undefined otherwise.*[3]

Theorem 1 (Characterization of externalized behavioral refinements). *Let* SP, SPI *be two specifications with signatures Σ, ΣI resp., let $\Sigma_{\textsc{COL}}$ be a* COL*-signature with underlying signature Σ and let $\kappa : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ be a constructor.*

$$\mathrm{SP}\ \overset{\Sigma_{\textsc{COL}}}{\rightsquigarrow^{\kappa}}\ \mathrm{SPI}\ \ \textit{if and only if}\ \ \mathrm{SP} \rightsquigarrow^{\kappa;\ \_\_/{\approx_{\Sigma_{\textsc{COL}}}}} \mathrm{SPI}.$$

Proof. The proof is a direct consequence of the definitions, in particular of the fact that $\kappa(\mathcal{M}od[\mathrm{SPI}]) \subseteq \mathrm{Beh}_{\Sigma_{\textsc{COL}}}(\mathcal{M}od[\mathrm{SP}])$ is equivalent to the inclusion $\kappa(\mathcal{M}od[\mathrm{SPI}])/{\approx_{\Sigma_{\textsc{COL}}}} \subseteq \mathcal{M}od[\mathrm{SP}]$. □

[2] The correctness proof is easy: First, the implementation indeed satisfies the non-observable equations (1) and (2) of SET due to the behavioral abstraction. That the reduct functor yields COL-algebras w.r.t. $\Sigma\textsc{Set}_{\textsc{COL}}$ follows from the observer complete form of the axioms; see [6] for more details.

[3] Obviously, $\_\_/{\approx_{\Sigma_{\textsc{COL}}}}$ is iso-preserving.

4 Behavioral Refinement: The Internalized View

The idea of the internalized view of behavioral refinement is to use a built-in behavioral semantics for specifications. For this purpose behavioral institutions which are tailored towards the behavioral aspects of system specifications provide an appropriate basis. Examples of such institutions are the framework of hidden algebra (see [10]) and the constructor-based observational logic institution COL (see [5]). In the following we will consider the COL institution for which no behavioral refinement concept has been investigated yet while for hidden algebra a refinement notion has been discussed in [13]. The COL institution has as signatures COL-signatures and as models COL-algebras as described in Section 2. COL-signature morphisms are standard signature morphisms which fulfill additional properties related to the preservation of constructor and observer operations and COL-morphisms between COL-algebras reflect behavioral relationships (see [5] for details). In particular, two Σ_{COL}-algebras A and B are Σ_{COL}-isomorphic if they are *observationally equivalent* (w.r.t. Σ_{COL}), i.e. if $A \equiv_{\Sigma_{\mathrm{COL}}} B$.

A crucial concept to obtain a built-in behavioral semantics is the *behavioral satisfaction relation*, denoted by $\models_{\Sigma_{\mathrm{COL}}}$, which generalizes the standard satisfaction relation of first-order logic by abstracting with respect to reachability and observability. From the reachability point of view, the valuations of variables are restricted to the elements of the Σ_{COL}-generated part $\mathrm{Gen}_{\Sigma_{\mathrm{COL}}}(A)$ only. From the observability point of view, the equality symbol "=" occurring in a first-order formula φ is not interpreted by the set-theoretic equality but by the observational equality $\approx_{\Sigma_{\mathrm{COL}},A}$ of elements.

In the following of this section we consider structured specifications over COL built by the specification building operations defined in Section 2.1. For instance, a basic COL specification $\mathrm{SP}_{\mathrm{COL}} = \langle \Sigma_{\mathrm{COL}}, \mathrm{Ax} \rangle$ consists of a COL-signature Σ_{COL} and a set Ax of Σ-sentences, called axioms. The semantics of $\mathrm{SP}_{\mathrm{COL}}$ is given by its signature Σ_{COL} and by its class of models

$$\mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}] = \{A \in \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}}) \mid A \models_{\Sigma_{\mathrm{COL}}} \mathrm{Ax}\}.$$

In order to define behavioral refinements for COL-specifications we can simply transfer the notions of implementation constructor and refinement used for the FOLEq institution in Definitions 1 and 2 to the COL institution. In particular, this means that COL-implementation constructors are required to preserve COL-isomorphisms, i.e. behavioral equivalences of algebras.

Definition 6 (COL-implementation constructor). *Let Σ_{COL}, ΣI_{COL} be two* COL*-signatures. A* COL-implementation constructor *from ΣI_{COL} to Σ_{COL} (also simply called a* COL-constructor*) is a partial function $\kappa_{\mathrm{COL}} : \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}}) \to \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}})$ which is* COL*-iso-preserving, i.e. for all $AI, BI \in \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}})$,*
if $AI \equiv_{\Sigma I_{\mathrm{COL}}} BI$ and $\kappa_{\mathrm{COL}}(AI)$ is defined
then $\kappa_{\mathrm{COL}}(BI)$ is defined and $\kappa_{\mathrm{COL}}(AI) \equiv_{\Sigma_{\mathrm{COL}}} \kappa_{\mathrm{COL}}(BI)$.
The definition domain of κ_{COL} is denoted by $Dom(\kappa_{\mathrm{COL}})$.

Definition 7 (Behavioral refinement: the internalized view). *Let* $\mathrm{SP}_{\mathrm{COL}}$, $\mathrm{SPI}_{\mathrm{COL}}$ *be two* COL*-specifications with signatures* Σ_{COL}, ΣI_{COL} *resp. and let* κ_{COL} *be a* COL*-constructor from* ΣI_{COL} *to* Σ_{COL}. $\mathrm{SPI}_{\mathrm{COL}}$ *is a* behavioral refinement *of* $\mathrm{SP}_{\mathrm{COL}}$ w.r.t. κ_{COL}, *denoted by* $\mathrm{SP}_{\mathrm{COL}} \rightsquigarrow^{\kappa_{\mathrm{COL}}} \mathrm{SPI}_{\mathrm{COL}}$, *if*

$$\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}] \subseteq Dom(\kappa_{\mathrm{COL}}) \text{ and } \kappa_{\mathrm{COL}}(\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}]) \subseteq \mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}].$$

An important question is, of course, which implementation constructors are appropriate for COL-refinements. As a first approach one could simply consider COL-signature morphisms $\sigma_{\mathrm{COL}} : \Sigma_{\mathrm{COL}} \to \Sigma I_{\mathrm{COL}}$. Since COL is an institution, the corresponding COL-reduct functor $\_|_{\sigma_{\mathrm{COL}}} : \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}}) \to \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}})$ preserves COL-isomorphisms, i.e. is a COL-implementation constructor. Hence it is tempting to consider COL-refinements where the syntactic relationship between the specification $\mathrm{SP}_{\mathrm{COL}}$ to be implemented and the implementing specification $\mathrm{SPI}_{\mathrm{COL}}$ is established by a COL-signature morphism. This approach has, however, a serious drawback because the implementing specification $\mathrm{SPI}_{\mathrm{COL}}$ usually has constructor and observer operations $\mathrm{OPI}_{\mathrm{Cons}}$, $\mathrm{OPI}_{\mathrm{Obs}}$ which are unrelated to the constructor and observer operations $\mathrm{OP}_{\mathrm{Cons}}$, $\mathrm{OP}_{\mathrm{Obs}}$ of the specification $\mathrm{SP}_{\mathrm{COL}}$ to be implemented. As a simple example we consider below the implementation of sets by lists where the observer for sets is the membership test *isin* while the observer operations for lists are, as usual, the *head* and *tail* operations. Hence the COL-specifications of sets and lists cannot be related by a COL-signature morphism which would require the preservation of constructor and observer operations.This is the reason why we want to consider standard signature morphisms and their reduct functors as implementation constructors for COL-specifications.

But before let us still point out that from a methodological point of view it is indeed adequate not to stick to COL-signature morphisms when we construct implementations. COL-signature morphisms are the appropriate tool to ensure encapsulation of COL-specifications (formally expressed by the satisfaction condition of an institution) which is indeed important when we construct large specifications in a modular way (often called *horizontal composition*). But when we discuss refinements and compositions of refinement steps (often called *vertical composition*) this is a totally different matter. Indeed, talking about encapsulation when relating abstract and concrete specifications makes no sense. An extensive discussion of this issue can also be found in [13].

Hence, let us consider two COL-specifications $\mathrm{SP}_{\mathrm{COL}}$, $\mathrm{SPI}_{\mathrm{COL}}$ with signatures Σ_{COL}, ΣI_{COL} resp. together with a (standard) signature morphism $\sigma : \Sigma \to \Sigma I$ (where Σ and ΣI are the underlying standard signatures of Σ_{COL} and ΣI_{COL} resp.). Moreover, let us consider the reduct functor $\_|_\sigma : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ as a partial function $\_|_\sigma : \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}}) \to \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}})$,[4] where

$\_|_\sigma(AI) \stackrel{\mathrm{def}}{=} AI|_\sigma$ if $AI|_\sigma$ is a Σ_{COL}-algebra,
$\_|_\sigma(AI)$ is undefined otherwise.

[4] By abuse of notation we use the same symbol $\_|_\sigma$ for the (total) reduct functor on $\mathrm{Alg}(\Sigma I)$ and for its induced partial reduct function on $\mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}})$.

The next lemma provides a simple criterion under which the (partial) reduct function on COL-algebras is COL-iso-preserving, i.e. is a COL-implementation constructor.

Lemma 1. *Let Σ_{COL}, ΣI_{COL} be* COL*-signatures with underlying signatures Σ, ΣI resp. Let $S_{\mathrm{Obs}}, SI_{\mathrm{Obs}}$ be the observable sorts and $S_{\mathrm{Loose}}, SI_{\mathrm{Loose}}$ be the loose sorts induced by Σ_{COL}, ΣI_{COL} resp. (see Section 2). If $\sigma(S_{\mathrm{Obs}}) \subseteq SI_{\mathrm{Obs}}$ and $\sigma(S_{\mathrm{Loose}}) \subseteq SI_{\mathrm{Loose}}$ then $\_\_|_\sigma : \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}}) \to \mathrm{Alg}_{\mathrm{COL}}(\Sigma_{\mathrm{COL}})$ is a* COL*-implementation constructor.*

Proof. We have to show that for all $AI, BI \in \mathrm{Alg}_{\mathrm{COL}}(\Sigma I_{\mathrm{COL}})$ the following holds:

1. If $AI \equiv_{\Sigma I_{\mathrm{COL}}} BI$ and $AI|_\sigma$ is a Σ_{COL}-algebra then $BI|_\sigma$ is a Σ_{COL}-algebra.
2. If $AI \equiv_{\Sigma I_{\mathrm{COL}}} BI$ then $AI|_\sigma \equiv_{\Sigma_{\mathrm{COL}}} BI|_\sigma$.

Proof of (1): Let $AI \equiv_{\Sigma I_{\mathrm{COL}}} BI$ and $AI|_\sigma$ be a Σ_{COL}-algebra. Then $AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}}$ iso $BI/{\approx_{\Sigma I_{\mathrm{COL}},BI}}$. Hence $(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}})|_\sigma$ iso $(BI/{\approx_{\Sigma I_{\mathrm{COL}},BI}})|_\sigma$. Due to the assumption $\sigma(S_{\mathrm{Obs}}) \subseteq SI_{\mathrm{Obs}}$ and $\sigma(S_{\mathrm{Loose}}) \subseteq SI_{\mathrm{Loose}}$, we can conclude that $AI|_\sigma$ is a Σ_{COL}-algebra iff $(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}})|_\sigma$ is a Σ_{COL}-algebra. Hence, $(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}})|_\sigma$ is a Σ_{COL}-algebra and so is $(BI/{\approx_{\Sigma I_{\mathrm{COL}},BI}})|_\sigma$. Again, by using the assumption, we conclude that $BI|_\sigma$ is a Σ_{COL}-algebra

Proof of (2): Let $In \stackrel{\mathrm{def}}{=} S_{\mathrm{Loose}}$, $Obs \stackrel{\mathrm{def}}{=} S_{\mathrm{Obs}}$, $InI \stackrel{\mathrm{def}}{=} SI_{\mathrm{Loose}}$, and $ObsI \stackrel{\mathrm{def}}{=} SI_{\mathrm{Obs}}$. Due to the assumption $\sigma(In) \subseteq InI$, $\sigma(Obs) \subseteq ObsI$ and according to [4] (Example 3.15), the reduct functor $\_\_|_\sigma : \mathrm{Alg}(\Sigma I) \to \mathrm{Alg}(\Sigma)$ is behavior respecting w.r.t. the partial observational equalities $\approx_{Obs,In}$ and $\approx_{ObsI,InI}$ in the sense of [4] (Def. 3.12). Since AI, BI are ΣI_{COL}-algebras, $\approx_{ObsI,InI,AI} = \approx_{\Sigma I_{\mathrm{COL}},AI}$ and $\approx_{ObsI,InI,BI} = \approx_{\Sigma I_{\mathrm{COL}},BI}$ and hence $AI \equiv_{\Sigma I_{\mathrm{COL}}} BI$ iff $AI \equiv_{ObI,InI} BI$. Since $\_\_|_\sigma$ is behavior respecting, $AI|_\sigma \equiv_{Obs,In} BI|_\sigma$. Since both reducts are Σ_{COL}-algebras this is equivalent to $AI|_\sigma \equiv_{\Sigma_{\mathrm{COL}}} BI|_\sigma$. □

Example 2. In contrast to Example 1 let us now consider COL-specifications of sets and lists. First, the COL-specification SETCOL of sets is given by including the observer *isin* into the COL-signature of the specification. (For simplicity, we do not consider constructor operations here.)

spec SETCOL =
  sorts *bool*, *nat*, *set*
  ops *true*, *false* : *bool*;
    0 : *nat*; *succ* : *nat* → *nat*; *plus* : *nat* × *nat* → *nat*;
    empty : *set*; *add* : *nat* × *set* → *set*;
    isin : *nat* × *set* → *bool*;
  observer (*isin*, *2*)
  axioms
  %% the same axioms as in SET (see Example 1)
end

The following specification LISTCOL provides a COL-specification of lists. As in any usual approach for a behavioral specification of lists we use the operations *head* and *tail* as observers for lists.

spec LISTCOL =
sorts *bool*, *nat*, *list*
ops *true*, *false* : *bool*;
0 : *nat*; *succ* : *nat* → *nat*; *plus* : *nat* × *nat* → *nat*;
empty : *list*; *cons* : *nat* × *list* → *list*;
head : *list* → *nat*; *tail* : *list* → *list*;
isin : *nat* × *list* → *bool*;
observers *head*, *tail*
axioms
%% the same axioms as in LIST (see Example 1)
end

For the implementation construction we use the same (standard) signature morphism $\sigma_{\text{SET}as\text{LIST}}$ as in Example 1 and the partial function

$$\_|_{\sigma_{\text{SET}as\text{LIST}}} : \text{Alg}_{\text{COL}}(\mathcal{S}ig[\text{LISTCOL}]) \to \text{Alg}_{\text{COL}}(\mathcal{S}ig[\text{SETCOL}])$$

induced by the reduct functor $\_|_{\sigma_{\text{SET}as\text{LIST}}}$ on standard algebras. It is important to note that the (image of the) observable sorts of SETCOL are included in the observable sorts of LISTCOL and hence, due to Lemma 1, the reduct functor is indeed a COL-implementation constructor, denoted by κ_{COL}. Thus we obtain the refinement relation SETCOL $\rightsquigarrow^{\kappa_{\text{COL}}}$ LISTCOL.[5]

5 Relating the Externalized and the Internalized Views of Behavioral Refinements

Let us first relate the implementation constructors used in the different approaches. Since any COL-algebra is also a (standard) algebra it is obvious that any implementation constructor $\kappa : \text{Alg}(\Sigma I) \to \text{Alg}(\Sigma)$ gives rise to a (partial) function $\kappa_{\text{COL}} : \text{Alg}_{\text{COL}}(\Sigma I_{\text{COL}}) \to \text{Alg}_{\text{COL}}(\Sigma_{\text{COL}})$ where

$\kappa_{\text{COL}}(AI) \stackrel{\text{def}}{=} \kappa(AI)$ if $\kappa(AI)$ is defined and is a Σ_{COL}-algebra,
$\kappa_{\text{COL}}(AI)$ is undefined otherwise.

If this partial function is COL-iso-preserving then κ_{COL} is a COL-implementation constructor *induced* by κ. In particular this means that κ is compatible with observational equivalences between COL-algebras, a property which is frequently used in the literature in different contexts having its origin in the notion of stability introduced by Schoett [20]. Thus constructors κ which induce COL-constructors will synonymously be called *stable* constructors. A criterion for the stability of reduct functors along standard signature morphisms has been provided in Lemma 1. The following lemma states a useful consequence of stable constructors.

[5] The correctness proof can be reduced to the proof of the refinement relation of Example 1 due to the forthcoming Theorem 2 which relates externalized and internalized views of behavioral refinements.

Lemma 2. *Let κ be a constructor from ΣI to Σ and κ_{COL} be a* COL*-constructor from ΣI_{COL} to Σ_{COL} induced by κ. Then, for any class $CI \subseteq \mathrm{Alg}(\Sigma I)$ of ΣI-algebras and for any iso-closed class $C \subseteq \mathrm{Alg}(\Sigma)$ of Σ-algebras, it holds:*
If $CI \subseteq Dom(\kappa)$ and $\kappa(CI) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$
then $\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(CI) \subseteq Dom(\kappa)$ and $\kappa(\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(CI)) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$.

Proof. Let $AI \in \mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(CI)$. Then $AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}} \in CI$ and, by assumption, $\kappa(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}}) \in \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$. Hence, in particular, $\kappa(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}})$ is a Σ_{COL}-algebra. Thus $\kappa_{\mathrm{COL}}(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}})$ is defined. Since $AI \equiv_{\Sigma I_{\mathrm{COL}}} AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}}$ and κ_{COL} is a COL-constructor, $\kappa_{\mathrm{COL}}(AI)$ is defined as well, i.e. $\kappa(AI)$ is a Σ_{COL}-algebra and thus $\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(CI) \subseteq Dom(\kappa)$.

Moreover, since κ_{COL} is COL-iso-preserving, $\kappa(AI) = \kappa_{\mathrm{COL}}(AI) \equiv_{\Sigma_{\mathrm{COL}}} \kappa_{\mathrm{COL}}(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}}) = \kappa(AI/{\approx_{\Sigma I_{\mathrm{COL}},AI}}) \in \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$. Since C is iso closed, $\mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$ is closed under COL-iso, i.e. under $\equiv_{\Sigma_{\mathrm{COL}}}$. Thus we obtain, as desired, $\kappa(AI) \in \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(C)$. □

From Lemma 2 we can easily conclude that for stable constructors, behavioral refinement steps according to the externalized view compose, i.e.

$$\mathrm{SP} \overset{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa} \mathrm{SPI},\ \mathrm{SPI} \overset{\Sigma I_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa'} \mathrm{SPI}' \text{ implies } \mathrm{SP} \overset{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa';\,\kappa} \mathrm{SPI}'.$$

Indeed, it has been pointed out already in [19] that the preservation of observational equivalences is crucial to guarantee vertical composition of so-called abstractor implementations which are a variant of the externalized approach. For the internalized approach, vertical composition is trivially guaranteed according to the built-in behavioral semantics which is used for both the specification to be implemented and for the implementing specification, i.e.

$$\mathrm{SP}_{\mathrm{COL}} \rightsquigarrow^{\kappa_{\mathrm{COL}}} \mathrm{SPI}_{\mathrm{COL}},\ \mathrm{SPI}_{\mathrm{COL}} \rightsquigarrow^{\kappa'_{\mathrm{COL}}} \mathrm{SPI}'_{\mathrm{COL}} \text{ implies } \mathrm{SP}_{\mathrm{COL}} \overset{\Sigma I_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa'_{\mathrm{COL}};\,\kappa_{\mathrm{COL}}} \mathrm{SPI}'_{\mathrm{COL}}.$$

In the following of this section we will show that under certain conditions (stability of constructors and behavioral closedness of specifications), externalized behavioral refinements and internalized behavioral refinements are expressible by each other. To relate the two approaches we first define a trivial syntactic translation $Forget_{\mathrm{COL}}$ from COL-specifications into standard specifications over FOLEq according to the structure of specifications:

$Forget_{\mathrm{COL}}(\langle \Sigma_{\mathrm{COL}}, \mathrm{Ax} \rangle) \overset{\mathrm{def}}{=} \langle \Sigma, \mathrm{Ax} \rangle$
where Σ is the underlying standard signature of Σ_{COL}

$Forget_{\mathrm{COL}}(\mathrm{SP}_{1,\mathrm{COL}} \cup \mathrm{SP}_{2,\mathrm{COL}}) \overset{\mathrm{def}}{=}$
$Forget_{\mathrm{COL}}(\mathrm{SP}_{1,\mathrm{COL}}) \cup Forget_{\mathrm{COL}}(\mathrm{SP}_{2,\mathrm{COL}})$

$Forget_{\mathrm{COL}}(\textbf{translate } \mathrm{SP}_{\mathrm{COL}} \textbf{ by } \sigma_{\mathrm{COL}}) \overset{\mathrm{def}}{=}$
translate $Forget_{\mathrm{COL}}(\mathrm{SP}_{\mathrm{COL}})$ **by** σ

where σ is the underlying standard signature morphism of σ_{COL}

$Forget_{\mathrm{COL}}$(**derive from** $\mathrm{SP}_{\mathrm{COL}}$ **by** σ_{COL}) $\stackrel{\mathrm{def}}{=}$
derive from $Forget_{\mathrm{COL}}(\mathrm{SP}_{\mathrm{COL}})$ **by** σ
where σ is the underlying standard signature morphism of σ_{COL}

We implicitly assume in the following that for any structured COL-specification $\mathrm{SP}_{\mathrm{COL}}$ its associated FOLEq-specification $Forget_{\mathrm{COL}}(\mathrm{SP}_{\mathrm{COL}})$ is denoted by SP and similarly for $\mathrm{SPI}_{\mathrm{COL}}$ etc. The following lemma states that COL-specifications and behavioral abstractions of their associated FOLEq-specifications are semantically equivalent.

Lemma 3. *Let* $\mathrm{SP}_{\mathrm{COL}}$ *be a* COL*-specification with signature* Σ_{COL} *and let* SP *be its associated* FOLEq*-specification. Assume that in the structured specification* SP*, each occurrence of the derive construct (if any) is applied to a behaviorally closed specification. Then* $\mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}] = \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$.[6]

Proof. The proof of the lemma is straightforward by induction on the structure of specifications. For the basic step we use the fact (see [5]) that for any Σ_{COL}-algebra A and Σ-sentence φ, $A \models_{\Sigma_{\mathrm{COL}}} \varphi$ iff $A/{\approx_{\Sigma_{\mathrm{COL}},A}} \models \varphi$ (where $\models$ denotes the standard satisfaction relation of first-order logic). The induction step for the union of two specifications is trivial and the induction steps for *translate* and *derive* utilize the fact that reduct functors w.r.t. COL-signature morphisms commute with black box constructions; see Theorem 51 of [5]. □

Theorem 2 (Relating externalized and internalized behavioral refinements). *Let* $\mathrm{SP}_{\mathrm{COL}}$, $\mathrm{SPI}_{\mathrm{COL}}$ *be two* COL*-specifications with signatures* Σ_{COL}, ΣI_{COL} *resp. and let* SP, SPI *be the associated* FOLEq*-specifications with signatures* Σ, ΣI *resp. Again we assume that in the structured specifications* SP *and* SPI*, each occurrence of the derive construct (if any) is applied to a behaviorally closed specification. Let* κ_{COL} *be a* COL*-constructor from* ΣI_{COL} *to* Σ_{COL} *induced by a constructor* κ *from* ΣI *to* Σ.

1. *If* $\mathrm{SP} \stackrel{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}^{\kappa} \mathrm{SPI}$ *then* $\mathrm{SP}_{\mathrm{COL}} \rightsquigarrow^{\kappa_{\mathrm{COL}}} \mathrm{SPI}_{\mathrm{COL}}$.
2. *If* SPI *is behaviorally closed w.r.t.* ΣI_{COL} *then* $\mathrm{SP} \stackrel{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}^{\kappa} \mathrm{SPI}$ *if and only if* $\mathrm{SP}_{\mathrm{COL}} \rightsquigarrow^{\kappa_{\mathrm{COL}}} \mathrm{SPI}_{\mathrm{COL}}$.

Proof. (1): By assumption,
$\mathcal{M}od[\mathrm{SPI}] \subseteq Dom(\kappa)$ and $\kappa(\mathcal{M}od[\mathrm{SPI}]) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$.
Hence, by Lemma 2 (and since $\mathcal{M}od[\mathrm{SP}]$ is iso-closed)
$\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}]) \subseteq Dom(\kappa)$ and
$\kappa(\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}])) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$.
Since, by Lemma 3,
$\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}] = \mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}])$ and $\mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}] = \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$
we obtain, as desired,

[6] In this equation the Σ_{COL}-algebras on the left-hand side are considered as standard Σ-algebras.

$\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}] \subseteq Dom(\kappa_{\mathrm{COL}})$ and $\kappa_{\mathrm{COL}}(\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}]) \subseteq \mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}]$.
(2): Conversely, if
$\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}] \subseteq Dom(\kappa_{\mathrm{COL}})$ and $\kappa_{\mathrm{COL}}(\mathcal{M}od[\mathrm{SPI}_{\mathrm{COL}}]) \subseteq \mathcal{M}od[\mathrm{SP}_{\mathrm{COL}}]$,
then we obtain, again by Lemma 3,
$\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}]) \subseteq Dom(\kappa)$ and
$\kappa(\mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}])) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$.
Since SPI is behaviorally closed w.r.t. ΣI_{COL}, $\mathcal{M}od[\mathrm{SPI}] \subseteq \mathrm{Beh}_{\Sigma I_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SPI}])$ and therefore $\mathcal{M}od[\mathrm{SPI}] \subseteq Dom(\kappa)$ and $\kappa(\mathcal{M}od[\mathrm{SPI}]) \subseteq \mathrm{Beh}_{\Sigma_{\mathrm{COL}}}(\mathcal{M}od[\mathrm{SP}])$. □

6 Conclusion

We have studied the relationships between externalized and internalized behavioral refinements which we believe is useful for further elaborations of behavioral refinement notions in the context of particular specification frameworks, like, e.g., the algebraic specification language CASL [2]. Indeed the essential results of our study, in particular the main theorem pointing out the equivalence of the external and the internal views of behavioral refinements (under certain assumptions), are in principle independent of the chosen formalism. Hence, it should be possible to generalize our results to a more abstract category-theoretic setting, e.g. along the lines of [14].

An important further issue concerns proof techniques to verify behavioral refinements. It seems that the most efficient way would be to reduce both, the externalized and the internalized notions, to the proof of refinement relations between standard first-order logic specifications (possibly involving sort generation constraints). Indeed Theorem 1 an 2 induce immediately the following two proof rules:

$$\frac{\mathrm{SP} \rightsquigarrow^{\kappa;\ \text{--}/\approx_{\Sigma_{\mathrm{COL}}}} \mathrm{SPI}}{\mathrm{SP} \overset{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa} \mathrm{SPI}}$$

$$\frac{\mathrm{SP} \overset{\Sigma_{\mathrm{COL}}}{\rightsquigarrow}{}^{\kappa} \mathrm{SPI}}{\mathrm{SP}_{\mathrm{COL}} \rightsquigarrow^{\kappa_{\mathrm{COL}}} \mathrm{SPI}_{\mathrm{COL}}}$$

Then, further proof rules are needed for proving $\mathrm{SP} \rightsquigarrow^{\kappa;\ \text{--}/\approx_{\Sigma_{\mathrm{COL}}}} \mathrm{SPI}$. A useful source for this purpose are the proof techniques for the validity of first-order sentences in behavioral quotient specifications provided in [3].

References

1. E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors. *Algebraic Foundations of Systems Specification*. Springer, 1999.
2. E. Astesiano, H. Kirchner M. Bidoit, B. Krieg-Brückner, P.D. Mosses, D.T. Sannella, and A. Tarlecki. CASL: The Common Algebraic Specification Language. *Theoretical Computer Science*, 286(2):153–196, 2002.

3. M. Bidoit, M.-V. Cengarle, and R. Hennicker. Proof systems for structured specifications and their refinements. In *[1]*, chapter 11, pages 385–433. Springer, 1999.
4. M. Bidoit and R. Hennicker. Modular correctness proofs of behavioural implementations. *Acta Informatica*, 35:951–1005, 1998.
5. M. Bidoit and R. Hennicker. Constructor-based observational logic. *Journal of Logic and Algebraic Programming*, 2005, to appear. Preliminary version available at `www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BID-HEN-JLAP.pdf`.
6. Michel Bidoit and Rolf Hennicker. Observer complete definitions are behaviourally coherent. In *Proc. OBJ/CafeOBJ/Maude Workshop at Formal Methods'99, Toulouse, France, Sep. 1999*, pages 83–94. THETA, 1999.
7. Michel Bidoit, Rolf Hennicker, and Martin Wirsing. Behavioural and abstractor specifications. *Science of Computer Programming*, 25(2–3):149–186, 1995.
8. H. Ehrig and H.-J. Kreowski. Refinement and implementation. In *[1]*, chapter 7, pages 201–242. Springer, 1999.
9. J. Goguen and J.A. Meseguer. Universal realization, persistent interconnection and implementation of abstract modules. In *Proc. ICALP'82*, volume 140 of *Lecture Notes in Computer Science*, pages 265–281. Springer, 1982.
10. J. Goguen and G. Roşu. Hiding more of hidden algebra. In J.M. Wing, J. Woodcock, and J. Davies, editors, *Proc. Formal Methods (FM'99)*, volume 1709 of *Lecture Notes in Computer Science*, pages 1704–1719. Springer, 1999.
11. Joseph Goguen and Rod Burstall. Institutions: abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146, 1992.
12. R. Hennicker. Observational implementation of algebraic specifications. *Acta Informatica*, 35:951–1005, 1998.
13. G. Malcolm and J. Goguen. Proving correctness of refinement and implementation. Technical Report PRG-114, Oxford University Computing Laboratory, 1994.
14. Michal Misiak. Behavioural semantics of algebraic specifications in arbitrary logical systems. In *Recent Trends in Algebraic Development Techniques*, volume 3423 of *LNCS*, pages 144–161. Springer, 2004.
15. M.P. Nivela and F. Orejas. Initial behaviour semantics for algebraic specifications. In *Recent Trends in Data Type Specification*, volume 332 of *LNCS*, pages 184–207. Springer, 1988.
16. F. Orejas, M. Navarro, and A. Sanchez. Implementation and behavioural equivalence. In *Recent Trends in Data Type Specification*, volume 655 of *Lecture Notes in Computer Science*, pages 93–125. Springer, 1993.
17. D. Sannella and A. Tarlecki. On observational equivalence and algebraic specification. *Journal of Computer and System Sciences*, 34:150–178, 1987.
18. Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.
19. D.T. Sannella and A. Tarlecki. Toward formal development of programs from algebraic specifications: implementation revisited. *Acta Informatica*, 25:233–281, 1988.
20. O. Schoett. Data abstraction and correctness of modular programming. Technical Report CST-42-87, University of Edinburgh, 1987.
21. Andrzej Tarlecki. Institutions: An Abstract Framework for Formal Specification. In *[1]*, chapter 4, pages 105–130. Springer, 1999.
22. Martin Wirsing. Algebraic Specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 13, pages 676–788. Elsevier Science Publishers B.V., 1990.