Refinement via interpretation

Manuel A. Martins[1], Alexandre Madeira[1,2] and L. S. Barbosa[2]
[1] Department of Mathematics, Aveiro University, Aveiro, Portugal
{martins, madeira}@ua.pt
[2]Department of Informatics & CCTC, Minho University, Portugal
lsb@di.uminho.pt

Abstract

Traditional notions of refinement of algebraic specifications, based on signature morphisms, are often too rigid to capture a number of relevant transformations in the context of software design, reuse and adaptation. This paper proposes an alternative notion of specification refinement, building on recent work on logic interpretation. The concept is discussed, its theory partially developed, its use illustrated through a number of examples.

Keywords: Algebraic specification, refinement, logic interpretation.

1. Introduction

The industrial demand for high-assurance software opens a window of opportunity for mathematically based development methods, able to design complex systems at ever-increasing levels of reliability and security. The aphorism *proofs pay the rent* witnesses this change, which is re-shaping our understanding of what Informatics is about, after all.

Algebraic specification methods [STar, Tar03, MT92, Wir90], having played a pioneer role in this process, constitute a large and mature body of knowledge and active research in the triple dimension of foundations, methodologies and applications. Central to such methods, namely to CASL [MHST03], its landmark realisation, is the process of *stepwise refinement* [ST88, BH06] through which a complex design is produced by incrementally adding details and reducing non determinism with respect to the original, high-level specifications. This is done step-by-step until the specification becomes a precise description of a concrete model, technically an algebra.

In the traditional framework of algebraic specification, *signature morphisms* are used to translate a specification into another one over a different signature [San00]. This enables the possibility to rename, add, remove and group together various signature components which is very useful during the specification and development processes. In a number of situations, however, transformations based in *signature morphisms* are too rigid to be useful. This is the case in the context of software reuse. But also the emergence of new computing paradigms in which software composition and adaptation becomes essentially dynamic and distributed [Fia04], entails the need for more flexible approaches to what is taken as a valid transformation of specifications (see, for example, [BSR04]).

This paper is a step in that direction. It started from a challenge: that of looking for transformations supported by mappings which need not to be morphisms. Multi-functions, *i.e.*, functions mapping an element to a set of elements, seemed a natural candidate. Unfortunately, in most cases, the price to be paid is rather high. In general it is no more possible to define the reduct of an algebra by these mappings, and consequently the traditional semantical treatment of the algebraic specification process does not apply.

The alternative put forward in this paper builds on top of recent works which apply tools and results from *abstract algebraic logic* to the specification of software systems (cf. [MP07]). The new concept of refinement proposed here, and referred to as *refinement via interpretation*, is based on *logic interpretation*, a central tool in the study of equivalence semantics (see, *e.g.*, [BP89, BP, BR03, Cze01]). A definition of *interpretation* can be found in [BP], where it is formulated, for k-logical systems, as $(k - l)$-mappings, translating a k-dimensional sentence into a set of l-dimensional sentences over the same signature. A paradigmatic example is the interpretation of the *classical propositional calculus* into the *equational theory of boolean algebras* (cf. [BP, Example 4.1.2]).

As a result we arrive at a very simple, but still quite expressive, notion of refinement. In particular, several crucial transformations in software development, *e.g.*, *data encapsulation* or *decomposition of operations into atomic transactions*, are captured in this framework. This increased flex-

ibility comes, essentially, from the possibility of mapping an equation into a set of equations, while signature morphisms map each formula into another one, preserving its structure.

The paper introduces this new concept, develops basic results in its theory and illustrates its relevance and applicability in a number of examples. Sections 3 and 4 contain the main results and examples. In order to keep exposition self-contained, section 2 provides the necessary background. This section can, naturally, be skipped by the informed reader. Finally, section 5 concludes and points out a few topics for future work.

2. Preliminaries

2.1. Universal (sorted) algebra

In this section, we recall some notions of *universal sorted algebra*. A detailed presentation of these concepts may be found in [STar] and [MT92].

Let S be a non empty set whose elements are called *sorts*. A *S-sorted set* is a S-indexed family of sets $A = (A_s)_{s \in S}$. A is *nonempty* if $A_s \neq \varnothing$ for each $s \in S$. We say that a S-sorted set A is *locally finite* (locally countable infinite) if, for any $s \in S$ A_s is a finite (countable infinite, resp.) set, and we say that A is *globally finite* if A is locally finite and $A_s = \emptyset$ except for a finite number of sorts (observe that if S is finite, then *local* implies *global* finiteness). The usual set operations are extended to many sorted componentwise (e.g., we say that $A \subseteq B$ if for any $s \in S$ $A_s \subseteq B_s$). The set of all S-sorted subsets of A is denoted by $\mathcal{P}(A)$ and the set of all globally finite S-sorted subsets of A by $\mathcal{PG}(A)$.

Given two S-sorted sets A and B, an *S-sorted mapping from A into B* is an S-sorted set $f = (f_s)_{s \in S}$ where $f_s : A_s \to B_s$; sometimes we write just f for the components f_s of f. An $S - S'$-sorted *multi-function* from a S-sorted set A to a S'-sorted set B, denoted by $\tau : A \to B$, is an S-family of mappings $(\tau_s : A_s \to \mathcal{P}(B))_{s \in S}$. τ is said to be *globally finite* if for each $s \in S$ and for all $a \in A$, $\tau_s(a)$ is globally finite.

In the sequel, if S is clear from the context, we may omit explicit reference to S and we just say "sorted..." instead of "S-sorted...".

Definition 1 (Signature) *A* signature Σ *is a pair* (S, Ω)*, where:*

- *S is a set (of sort names);*
- *Ω is a $(S^* \times S)$-sorted set (of operation names),*

where S^* is the set of all the finite sequences of S elements.

The symbols in $\Omega_{s_1 \ldots s_n, s}$, $n \geq 1$, are called *operation symbols* with *arguments of sort* $s_1, ..., s_n$ and *range sort* s. The elements of $\Omega_{\epsilon, s}$, $s \in S$, are called *constants of sort* s (ϵ denotes the empty sequence).

We write $f : s_1, \ldots, s_n \to s \in \Sigma$ to mean that $s_1, ..., s_n, s \in S$ and $f \in \Omega_{s_1 \ldots s_n, s}$; $s_1, \ldots, s_n \to s$ is the *type of* f.

Definition 2 (Σ-algebra) *Let $\Sigma = (S, \Omega)$ be a signature. A* Σ-algebra A *consists of*

- *an S-sorted set $A = (A_s)_{s \in S}$, where for all $s \in S$ $A_s \neq \emptyset$;*
- *for any $f \in \Omega_{s_1 \ldots s_n, s}$, a function $f^A : A_{s_1} \times \cdots \times A_{s_n} \to A_s$;*

A is called the carrier set *of the algebra. We will denote as usually the algebra and its carrier set with the same roman letter.*

An *homomorphism* between two Σ-algebras A and B is an S-sorted mapping $h : A \to B$ between the correspondent carrier sets, satisfying for each $f : s_1, \ldots, s_n \to s \in \Sigma$, $h_s(f^A(a_1, \ldots, a_n)) = f^B(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$ for all $a_i \in A_{S_i}, 1 \leq i \leq n$. An injective homomorphism f is called a *monomorphism*. If f is surjective, it is called an *epimorphism*. We say that h is an *isomorphism* if it is both an injective and a surjective homomorphism.

A set of *variables* for a signature $\Sigma = (S, \Omega)$ is a nonempty S-sorted set X of pairwise disjoint sets. The elements in X_s are called *s-variables*. To denote that a variable x is of sort s we write $x{:}s$. Throughout the paper we take the usual assumption that the variables and the symbols in Ω have different denotations.

Definition 3 ($T_\Sigma(X)$) *Let Σ be a signature and X a set of variables for Σ. The set of Σ-terms in the variables X is the smallest set $T_\Sigma(X)$ such that:*

- *for any sort $s \in S$, $X_s \subseteq (T_\Sigma(X))_s$;*
- *if $f :\to s \in \Sigma$ then $f \in (T_\Sigma(X))_s$;*
- *if $f : s_1, \ldots, s_n \to s \in \Sigma$ and $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$, then $f(t_1, \ldots, t_n) \in T_\Sigma(X)_s$;*

The elements in $T_\Sigma(X)_s$ are called *Σ-terms of sort s*. We write $t(x_1, \ldots, x_n)$ to mean that the variables which occur in t are among $x_1, \ldots, x_n$. The terms without variables are called *ground terms*. A signature is said to be standard if there is a ground term for every sort S.

It is well know that we can define, in a standard way, operations over $T_\Sigma(X)$ to obtain a Σ-algebra which is called *term algebra over* Σ. An endomorphism $\sigma : T_\Sigma(X) \to T_\Sigma(X)$ is called a *substitution*.

Definition 4 (Assignment) *Let $\Sigma = (S, \Omega)$ be a signature, X be a set of variables for Σ and A be a Σ-algebra. An* assignment $h : X \to A$ *is a S-family of mappings*

$(h_s : X_s \to A_s)_{s\in S}$. *Any assignment* h *uniquely extends to a* Σ*-homomorphism* $\bar{h} : T_\Sigma(X) \to A$ *as follows:*

- $\bar{h}_s(x) =_{def} h_s(x), x \in X_s$;
- *for any* $f : s_1, \ldots, s_n \to s \in \Sigma$, *for all* $t_1 \in T_\Sigma(X)_{s_1}, \ldots, t_n \in T_\Sigma(X)_{s_n}$, $\bar{h}_s(f(t_1, \ldots, t_n)) =_{def} f^A(\bar{h}_{s_1}(t_1), \ldots, \bar{h}_{s_n}(t_n))$.

In the sequel we will simply write h *instead of* $\bar{h}$.

Given a term $t(x_1, \ldots, x_n)$ and a assignment $h : X \to A$ such that $h(x_i) = a_i$, $1 \leq i \leq n$, we denote $h(t)$ by $t^A(a_1, \ldots, a_n)$ (in the sequel the superscript A may be omitted).

In the remaining of this section, Σ is a signature and X is a set of variables for Σ. A Σ*-equation of sort* S *over* X (equation for short, if Σ and X are clear from the context) is a pair $\langle t, t'\rangle$, where $t, t' \in T_\Sigma(X)_s$ for some $s \in S$. Usually, we represent a Σ-equation $\langle t, t'\rangle$ by $t \approx t'$. We denote the S-sorted set of all Σ-equations (over X) by $\mathrm{Eq}_\Sigma(X)$.

Since the components of the family $\mathrm{Eq}_\Sigma(X)$ are pairwise disjoint, an S-sorted subset Γ of $\mathrm{Eq}_\Sigma(X)$ will be identified with the unsorted set $\bigcup_{s\in S} \Gamma_S$; of course this applies to $\mathrm{Eq}_\Sigma(X)$ itself.

A Σ*-conditional equation over* X (a conditional equation for short, if Σ and X are clear from the context) is a pair $\langle \Gamma, e\rangle$ where Γ is a globally finite subset of $\mathrm{Eq}_\Sigma(X)$ and $e \in \mathrm{Eq}_\Sigma(X)$. A conditional equation $\langle \{t_1 \approx t'_1, \ldots, t_n \approx t'_n\}, t \approx t'\rangle$ will be written as $t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t'$. The equations in Γ are called *premisses* and $t \approx t'$ the *conclusion*. An equation may be seen as a conditional equation without premisses, hence we will identify the equation $t \approx t'$ with the conditional equation $\langle \emptyset, t \approx t'\rangle$. The set of all Σ-conditional equations is denoted by $\mathrm{Ceq}_\Sigma(X)$ (including equations).

Let A be a Σ-algebra, and $t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t' \in \mathrm{Ceq}_\Sigma(X)$. The Σ-algebra A *is a model of the* Σ*-conditional equation* $t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t'$, in symbols $A \models t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t'$, if for every valuation $h : X \to A$, $h(t_i) = h(t'_i)$ for every $i \leq n$ implies $h(t) = h(t')$. For the special case of equations, $A \models t \approx t'$ if for every valuation $h : X \to A$ $h(t) = h(t')$. Given a set Φ of conditional equations, we write $A \models \Phi$ if $A \models \xi$ for every $\xi \in \Phi$.

2.1.1. Signature morphisms Signature morphisms are important tools on the implementation procedure of software systems.

Definition 5 (Signature morphism) *Let* $\Sigma = (S, \Omega)$ *and* $\Sigma' = (S', \Omega')$ *be signatures. A* signature morphism $\sigma : \Sigma \to \Sigma'$, *is a pair* $\sigma = (\sigma_{sorts}, \sigma_{op})$, *where* $\sigma_{sorts} : S \to S'$ *and* $\sigma_{op} : \Omega \to \Omega'$ *is a* $(S^* \times S)$*-family of functions respecting the sorts of operations names in* Ω, *that is,* $\sigma_{op} = (\sigma_{\omega,s} : \Omega_{\omega,s} \to \Omega'_{\sigma^*_{sorts}(\omega),\sigma_{sorts}(s)})_{\omega\in S^*,s\in S}$ *(where for* $\omega = s_1 \ldots s_n \in S^*, \sigma^*_{sorts}(\omega) = \sigma_{sorts}(s_1) \ldots \sigma_{sorts}(s_n)$*).*

Definition 6 (Reduct Algebra) *Let* A' *be a* Σ'*-algebra, and* $\sigma : \Sigma \to \Sigma'$ *be a signature morphism. The* σ-reduct *of* A' *is the* Σ*-algebra* $A' \restriction_\sigma$ *defined as follows:*

- *for any* $s \in S$, $(A' \restriction_\sigma)_s = A'_{\sigma(s)}$, *and*
- *for all* $f : s_1, \ldots, s_n \to s \in \Sigma$, $f^{A'\restriction_\sigma} = \sigma op(f)^{A'}$.

To extend signature morphisms to terms we have to take care about the set of variables we are dealing with. Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism with $\Sigma = (S, \Omega)$, $\Sigma' = (S', \Omega')$. Given a set of variables $X = (X_s)_{s\in S}$ for Σ, we consider a set of variables X' for Σ' such that for any $s' \in S'$, $X'_{s'} = \bigcup_{\sigma(s)=s'} X_s$ (cf. [STar, Mar06]).

Therefore, we can extend, in an uniquely way, σ (more precisely σ together with the inclusion on the variables) to a homomorphism from $T_\Sigma(X)$ into $T_{\Sigma'}(X')$ that we will denote by σ too (see [STar]). Given a equation $t \approx t'$, we write $\sigma(t \approx t')$ for $\sigma(t) \approx \sigma(t')$.

The following well known result is the basis of the traditional refinement procedure.

Lemma 1 (Satisfaction Lemma [GB92]) *Let* Σ *and* Σ' *be signatures,* $\sigma : \Sigma \to \Sigma'$ *a signature morphism,* A' *a* Σ'*-algebra and* ξ *a conditional equation. Then,*

$$A' \models \sigma(\xi) \textit{ iff } A' \restriction_\sigma \models \xi.$$

2.2. Specification logic

Given a signature Σ, a locally countable infinite S-sorted set X of variables for Σ and a class K of Σ-algebras, the *equational consequence relation* determined by K w.r.t X is the relation between sets of Σ-equations and individual Σ-equations in $\mathrm{Eq}_\Sigma(X)$ defined by

$$\{t_i \approx t'_i | i \in I\} \models_K t \approx t' \text{ if,}$$

for all $A \in K$ and every valuation $\alpha : X \to A$,

$$\alpha(t_i) = \alpha(t'_i) \text{ for every } i \in I \text{ implies } \alpha(t) = \alpha(t').$$

The relation $\models_K$ is said to be *finitary* if $\{t_i \approx t'_i | i \in I\} \models_K t \approx t'$ implies $\{t_i \approx t'_i | i \in J\} \models_K t \approx t'$ for some finite $J \subseteq I$.

Let A a Σ-algebra, and $\xi = t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t' \in \mathrm{Ceq}_\Sigma(X)$. It is easy to see that A is a model of ξ if $\{t_i \approx t'_i | i \leq n\} \models_{\{A\}} t \approx t'$.

Note that if I is finite, $\{t_i \approx t'_i | i \in I\} \models_K t \approx t'$ if and only if for every $A \in K$ $A \models t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t'$.

The relation $\models_K$ is a Tarski consequence relation on the set of equations, that is, it satisfies the following conditions for all $\Gamma, \Delta \subseteq \mathrm{Eq}_\Sigma(X)$ and $t \approx t' \in \mathrm{Eq}_\Sigma(X)$

1. $t \approx t' \in \Gamma$ implies $\Gamma \models_K t \approx t'$,
2. $\Gamma \models_K t \approx t'$ and $\Gamma \subseteq \Delta$ imply $\Delta \models_K t \approx t'$,
3. $\Gamma \models_K t \approx t'$ and $\Delta \models_K u \approx u'$ for every $u \approx u' \in \Gamma$ imply $\Delta \models_K t \approx t'$,
4. $\Gamma \models_K t \approx t'$ implies $\sigma(\Gamma) \models_K \sigma(t \approx t')$ for every substitution σ.

It can be proved that, if K is a class of Σ-algebras axiomatized by a set of Σ-conditional equations then the relation $\models_K$ is finitary (cf.[BR03] for the one-sorted case). In this case the relation can be defined in the Hilbert style by considering the set of Σ-equations in Φ together with the reflexivity axioms as the set of axioms, and the Σ-conditional equations in Φ together with the symmetry, transitivity and congruence rules as the inference rules. This is established in the following proposition.

Proposition 1 [BR03] *Let $\Gamma \cup \{t \approx t'\} \subseteq \mathrm{Eq}_\Sigma(X)$ and K be the class of Σ-algebras axiomatised by Φ. We have that $\Gamma \models_K t \approx t'$ if and only if, there is a finite sequence of equations $t_1 \approx t'_1, \ldots, t_n \approx t'_n$, called* a proof of $t \approx t'$ from Γ in $\models_K$, *such that $t_n \approx t'_n$ is $t \approx t'$ and for every $i = 1, \ldots, n$ one of the following conditions holds:*

1. *$t_i \approx t'_i \in \Gamma$;*
2. *there is an axiom $v \approx v'$ and a substitution σ such that $t_i \approx t'_i$ is $\sigma(v \approx v')$;*
3. *there is an inference rule $v_1 \approx v'_1 \wedge \cdots \wedge v_m \approx v'_m \rightarrow v \approx v'$, and a substitution σ such that $t_i \approx t'_i$ is $\sigma(v \approx v')$ and $\{\sigma(v_l \approx v'_l) | l < m\} \subseteq \{t_j \approx t'_j | j < i\}$.*

2.3. Algebraic specification

To specify a software system, we should define an adequate signature, taking in account the sorts and functions of the intended system, and we express, in an appropriated logical system, the desired functional behaviour of the signature operations.

An *algebraic specification* SP is a pair $\langle \Sigma, [\![SP]\!] \rangle$ where Σ is a signature, denoted by $Sig(SP)$ and $[\![SP]\!]$ is a class of Σ-algebras. This class of Σ-algebras is called the *model class of* SP, and an individual $Sig(SP)$-algebra in $[\![SP]\!]$ a *model of* SP. If ξ is the conditional equation $\langle \Gamma, e \rangle$, we write $SP \models \xi$ for $\Gamma \models_{[\![SP]\!]} e$. We say that a specification SP is *X-flat* if there is a sorted set of variables X for Σ and a set $\Phi \subseteq \mathrm{Ceq}_\Sigma(X)$ such that $[\![SP]\!] = \{A \in Alg(\Sigma) | A \models \Phi\}$. We represent an axiomatised specification $SP = \langle \Sigma, [\![SP]\!] \rangle$ as a pair $SP = \langle \Sigma, \Phi \rangle$ omitting explicit reference to the variables X; X is assumed to be a set of variables for Σ such that $\Phi \subseteq \mathrm{Ceq}_\Sigma(X)$ and $[\![SP]\!] = \{A \in Alg(\Sigma) | A \models \Phi\}$, where $Alg(\Sigma)$ denotes the call of all Σ-algebras. When Φ is a set of equations, the specification $SP = \langle \Sigma, \Phi \rangle$ is called an *equational specification*.

2.4. Refinement

Given a specification SP of a software system, the implementation process consists in building a correct realization (a program) of SP, i.e., built an algebra P such that $P \in [\![SP]\!]$, or at least a class of $Sig(SP)$-algebras SP' such that $[\![SP']\!] \subseteq [\![SP]\!]$, and with SP' small enough for the desired work. During this process, we enrich SP with implementation decisions, in order to obtain a complete description of the intended program (desired algebra).

The *stepwise refinement process* (see [STar, ST97, Mar06]) is the systematic process by which, from an initial abstract specification SP_0 more concrete specifications are built by introducing new requirements leading to

$$SP_0 \rightsquigarrow SP_1 \rightsquigarrow SP_2 \rightsquigarrow \cdots \rightsquigarrow SP_{n-1} \rightsquigarrow SP_n,$$

where for all $1 \leq i \leq n$, $SP_{i-1} \rightsquigarrow SP_i$ means $[\![SP_i]\!] \subseteq [\![SP_{i-1}]\!]$. Each step of this process is called a *refinement step* or simply a *refinement*. Note that if $SP \rightsquigarrow SP'$ and $SP' \rightsquigarrow SP''$ then $SP \rightsquigarrow SP''$, since $Sig(SP) = Sig(SP') = Sig(SP'')$ and $[\![SP'']\!] \subseteq [\![SP']\!] \subseteq [\![SP]\!]$. This transitivity, named *vertical composition*, assure that $SP_0 \rightsquigarrow SP_n$.

The introduction of new requirements mentioned above is achieved through the specification of suitable signature morphisms. Supported on the *Satisfaction Lemma* (Lemma 1), we have the following generalisation of the refinement concept:

Definition 7 (σ-Refinement) *Let $\sigma : \Sigma \rightarrow \Sigma'$ be a signature morphism. The specification SP' over Σ' is a* σ-refinement of SP, *in symbols $SP \rightsquigarrow_\sigma SP'$, if*

$$[\![SP']\!] \restriction_\sigma \subseteq [\![SP]\!],$$

where $[\![SP']\!] \restriction_\sigma = \{A \restriction_\sigma | A \in [\![SP']\!]\}$.

Note that a refinement is just a id-refinement with id the identity morphism. Since the composition of two signature morphisms is a signature morphism, we have, by *Satisfaction Lemma* (Lemma 1), that the vertical composition holds. I.e., if $SP_0 \rightsquigarrow_{\sigma_1} SP_1$ and $SP_1 \rightsquigarrow_{\sigma_2} SP_2$ we have $SP_0 \rightsquigarrow_{\sigma_2 \circ \sigma_1} SP_2$, and for the case with n steps, we have $SP_0 \rightsquigarrow_{\sigma_n \circ \cdots \circ \sigma_1} SP_n$.

It follows an important characterisation of σ-refinements.

Theorem 1 *Let $\sigma : \Sigma \rightarrow \Sigma'$ be a signature morphism, $SP = \langle \Sigma, \Phi \rangle$ a X-flat specification and SP' a specifications over Σ'. Then, $SP \rightsquigarrow_\sigma SP'$ iff $SP' \models \sigma(\Phi)$.*

3. Refinement via interpretation

This section introduces, exemplifies and discusses the concept of *refinement via interpretation* — the core con-

tribution of this paper. As mentioned in the Introduction, this new perspective on algebraic specification refinement is based on the notion of *logic interpretation*, a central concept in the abstract algebraic theory of deductive systems [BP89, BP, BR03, Cze01]. In such a context, an *interpretation* is a particular kind of *translation*, both notions being discussed in the next two sub-sections. Then, sub-section 3.3 introduces our notion of refinement and illustrates its suitability through a number of examples. Finally, its theory is (partially) developed in sub-section 3.4.

3.1. Translations

A number of notions of translation between logical systems have been proposed in the literature (see, for example, [Fei97, FD01, BP, MDT09]). In the sequel we adopt the following definition, assuming that all sets of variables are locally countable infinite.

Definition 8 (Translation) *Let $\Sigma = (S, \Omega)$, $\Sigma' = (S', \Omega')$ be two signatures, and X and X' sets of variables for Σ and Σ' respectively. A* translation from Σ to Σ' with respect to (w.r.t.) the set of variables X and X' *is a globally finite $S - S'$-sorted* multi-function *from* $\mathrm{Eq}_\Sigma(X)$ *to* $\mathrm{Eq}_{\Sigma'}(X')$.

When $\Sigma = \Sigma'$ and $X = X'$, we call τ a *self translation of Σ w.r.t the set of variables X*. And, in this case, we say that τ *commutes with substitutions* if for every substitution s and every equation $e \in \mathrm{Eq}_\Sigma(X)$ $\tau(s(e)) = s(\tau(e))$.

We extend, in a natural way, a translation $\tau : \mathrm{Eq}_\Sigma(X) \to \mathrm{Eq}_{\Sigma'}(X')$ to a multi-function $\tau^* : \mathrm{Ceq}_\Sigma(X) \to \mathrm{Ceq}_{\Sigma'}(X')$ as follows:

$$\tau^*(\xi) = \{\langle \bigcup_{t \approx t' \in \Gamma} \tau(t \approx t'), e' \rangle : e' \in \tau(e)\}.$$

for any conditional equation $\xi = \langle \Gamma, e \rangle$. In the sequel, we will identify τ^* with τ. Notice how the reason for requiring $\tau_s(x)$ to be global finite becomes clear from the definition of τ^*. We may now prove the following result.

Proposition 2 *Let $\Sigma = (S, \Omega)$ be a standard signature, X a set of variables for Σ and τ a self translation of Σ w.r.t X. Then the following conditions are equivalent:*

1. *τ commutes with substitutions.*
2. *There exists an S-sorted set of equations $E(x, y) \subseteq \mathrm{Eq}_\Sigma(X)$ such that, for any $t \approx t' \in \mathrm{Eq}_\Sigma(X)_s$, $\tau_s(t \approx t') = E_s(t, t')$.*

Proof. Assume τ commutes with arbitrary substitutions. We fix distinct variables p, q and we define $E(p, q) := \tau(p \approx q)$. Suppose $\mathrm{Var}(E(p, q)) \subseteq \{p, q, r_1, r_2, \dots\}$. Let e be a substitution such that $e(p) = p$, $e(q) = q$ and $e(r_i) = p$ for all $i \in I$. By assumption, $E(p, q, p, \dots) = e(E(p, q)) = e\big(\tau(p \approx q)\big) = \tau\big(e(p) \approx e(q)\big) = \tau(p \approx q) = E(p, q, r_1, r_2, \dots)$. Hence, $\{r_1, r_2, \dots\} \subseteq \{p, q\}$. Thus $\mathrm{Var}(E(p, q)) \subseteq \{p, q\}$. Now, let $\varphi \approx \psi \in \mathrm{Eq}$ and e a substitution such that $e(p) = \varphi$ and $e(q) = \psi$. We have that $\tau(\varphi \approx \psi) = \tau\big(e(p) \approx e(q)\big) = e\big(\tau(p \approx q)\big) = e\big(E(p, q)\big) = E\big(e(p), e(q)\big) = E(\varphi, \psi)$.

Suppose now that 2 holds. Let α be a substitution in Σ. Then, for any $t \approx t' \in \Sigma$,

$$\begin{aligned}\alpha(\tau(t \approx t')) =& \alpha(E_s(t, t')) = E_s(\alpha(t), \alpha(t'))) \\ &= \tau(\alpha(t) \approx \alpha(t')) = \tau(\alpha(t \approx t'))\end{aligned}$$

□

3.2. Interpretations

Defined as a multi-function, a translation maps a term into a set of terms, and this is what makes translations interesting to establish relationships between specification. Recall that, on the other hand, a signature morphism maps a term into just another term.

Not all translations, however, are suitable to capture the meaning of interpreting a specification into another one. The following definition singles out the relevant ones:

Definition 9 (Interpretation) *Let τ be a translation from Σ to Σ' w.r.t. the set of variables X and X'. Let SP be a specification over Σ. We say that τ* interprets *SP if there is a specification SP' over Σ' such that, for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi$ if and only if $SP' \models \tau(\xi)$. In this case we say that τ* interprets *SP in SP' and SP'* is a *τ*-interpretation of *SP*.

Example 1 *The interpretation of the class HA of* Heyting algebras *in the class BA of (the specification of)* Boolean algebras *is a classical example of an interpretation. Let X be a numerable set of variables and Σ the usual signature for booleana algebras (and for Heyting algebras). Consider the well known* double negation (propositional) translation [1]*:*

$$\begin{array}{rccc} \iota : & T_\Sigma(X) & \to & T_\Sigma(X) \\ & t & \mapsto & \iota(t) = \neg\neg t \end{array}$$

Let τ be the a self translation of Σ w.r.t X defined by

$$\tau(t \approx t') = \{\iota(t) \approx \iota(t')\}.$$

It can be shown that τ interprets the specification BA of the boolean algebras in the specification HA of the Heyting algebras (cf. [BR03]).

It is not difficult to see that

1 This translation is known by Glivenko's interpretation. It establishes a strict relationship between classical derivability and intuitionistic derivability, namely, φ is a theorem in CPC then $\neg\neg\varphi$ is a theorem in IPC (cf. [BR03]). The result establishes the bridge between the algebraic semantics of the *classical propositional calculus* and the algebraic semantics of *intuitionistic propositional calculus*.

Theorem 2 *Let SP and SP' be two algebraic specifications over a signature Σ and τ a recursive self translation of Σ w.r.t to a set of variables X that commutes with arbitrary substitutions and interprets SP in SP'. If SP is decidable then SP' is decidable.*

Definition 10 (τ-model) *Let τ be a translation from Σ to Σ' w.r.t. the set of variables X and X', and SP a specification over Σ. A Σ'-algebra A' is a* τ-model of SP *if for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi$ implies $A' \models \tau(\xi)$. We define the* τ-model class of SP, *denoted by $\mathrm{Mod}_\tau(SP)$, as the class of all τ-models of SP. We will denote the specification over Σ' whose models are $\mathrm{Mod}_\tau(SP)$ by SP^τ.*

Observe that for all conditional equation ξ, $SP \models \xi \rightarrow SP^\tau \models \tau(\xi)$.

Theorem 3 *Let τ be a translation from Σ to Σ' w.r.t. the set of variables X and X', and SP a specification over Σ. If τ interprets SP, then the specification SP^τ is the largest τ-interpretation of SP, i.e., with the largest class of models.*

Proof. Suppose that τ interprets SP. Let SP' be a specification that is a τ-interpretation of SP. Then for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi$ if and only if $SP' \models \tau(\xi)$. Hence all models of SP' are τ-models of SP. Thus, $[\![SP']\!] \subseteq [\![SP^\tau]\!]$.

So, we only need to prove that SP^τ is a τ-interpretation of SP. Let $\xi \in \mathrm{Ceq}_\Sigma(X)$. It is clear that $SP \models \xi$ implies $SP^\tau \models \tau(\xi)$. Suppose now that $SP^\tau \models \tau(\xi)$. Let SP' be a specification that is a τ-interpretation of SP (it exists since τ interprets SP). Since, $[\![SP']\!] \subseteq [\![SP^\tau]\!]$, $SP' \models \tau(\xi)$. Since SP' is a τ-interpretation of SP, $SP \models \xi$. □

Next theorem states that SP^τ is finitely axiomatized whenever SP is:

Theorem 4 *Let τ be a self translation of Σ w.r.t X and $SP = \langle \Sigma, \Phi \rangle$ be a X-flat specification. If τ commutes with substitutions then SP^τ is also X-flat and $SP^\tau = \langle \Sigma', \tau(\Phi) \rangle$. Moreover, if Φ is finite then SP^τ is finitely axiomatisable, i.e., X-flat.*

Proof. On the one hand, we have that for any $A' \in [\![SP^\tau]\!]$ and for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi$ implies $A' \models \tau(\xi)$. In particular, since $SP \models \Phi$, we have that $SP^\tau \models \tau(\Phi)$ and hence, $[\![SP^\tau]\!] \subseteq [\![\langle \Sigma', \tau(\Phi) \rangle]\!]$.

On the other hand, let $A \in \langle \Sigma', \tau(\Phi) \rangle$ and $\xi = \langle \Gamma, e \rangle$ be a conditional equation over X such that $SP \models \xi$ (i.e., $\Gamma \models_{[\![SP]\!]} e$). Then, it can be proved by induction on the length of a proof of e from Γ in $\models_{[\![SP]\!]}$ that $A \models \tau(\xi)$. Therefore A is a τ-model of SP and, so, $[\![\langle \Sigma', \tau(\Phi) \rangle]\!] \subseteq SP^\tau$.

Clearly, if Φ is finite then $\tau(\Phi)$ is also finite. Moreover, one can prove that $\tau(\Phi)$ constitutes an axiomatization for SP^τ. □

3.3. Refinement via interpretation

Logic interpretations provide the basic tool for the following definition:

Definition 11 (Refinement via interpretation) *Let SP be a specification over Σ and τ a translation from Σ to Σ' w.r.t. the set of variables X and X' which interprets SP. We say that a specification SP' over Σ'* refines the specification SP via the interpretation τ, *in symbols $SP \rightarrow_\tau SP'$, if for any $\xi \in \mathrm{Ceq}_\Sigma(X)$*

$$SP \models \xi \;\Rightarrow\; SP' \models \tau(\xi)$$

Let us now discuss some examples of refinements via interpretations. The first one is mainly of theoretical interest: it shows how (a specification of) an Heyting algebra can be regarded as a refinement of (a specification of) a Boolean algebra.

Example 2 *Consider the following specifications of* Boolean *and* Heyting algebras*:*

SPEC $BOOL$ = enrich $DIST-LATTICE$ by
[AX]
(1) $p \vee \neg p \approx \mathsf{tt}$
(2) $p \wedge \neg p \approx \mathsf{ff}$
(3) $p \vee \mathsf{tt} \approx \mathsf{tt}$
(4) $p \wedge \mathsf{ff} \approx \mathsf{ff}$

and

SPEC $HEYTING$ = enrich $DIST-LATTICE$ by
[GEN]
bool
[OP]
$\mathsf{tt} : \longrightarrow bool$
$\mathsf{ff} : \longrightarrow bool$
$\mathsf{neg} : bool \longrightarrow bool$
$\wedge : bool \times bool \longrightarrow bool$
$\vee : bool \times bool \longrightarrow bool$
$\Rightarrow : bool \times bool \longrightarrow bool$
[AX]
(1) $p \vee \mathsf{tt} \approx \mathsf{tt}$
(2) $p \wedge \mathsf{ff} \approx \mathsf{ff}$
(3) $p \Rightarrow p \approx \mathsf{tt}$
(4) $p \Rightarrow (q \vee r) \approx (p \Rightarrow q)$
(5) $(p \Rightarrow q) \wedge q \approx q$
(6) $p \wedge (p \Rightarrow q) \approx p \wedge q$
(7) $p \Rightarrow (q \wedge r) \approx (p \Rightarrow q) \wedge (p \Rightarrow r)$
(8) $(p \wedge q) \Rightarrow r \approx (p \Rightarrow r) \wedge (q \Rightarrow r)$

where $DIST-LATTICE$ is the specification of the distributive lattices (cf. [BS81]).

As in Example 1, multi-function τ defined by

$$\tau(t \approx t') \;=\; \{\neg\neg t \approx \neg\neg t'\}$$

interprets $BOOL$ in $HEYTING$. To show that $BOOL \rightarrow_\tau HEYTING$ just observe that for any axiom φ of $BOOL$, $HEYTING \models \tau(\varphi)$.

Our next example, although quite elementary, illustrates a key point. It shows how refinement via interpretation is able to capture *data encapsulation*, *i.e.*, the process of hiding a specific sort in a specification. This is a relevant issue in algebraic specification, in particular when the implementation target is an object-oriented framework: hidden sorts become the state space of object implementations, as discussed in, *e.g.*, [Fav98, DD05]. It is interesting to have such sort of transformations integrated in a stepwise refinement process.

Example 3 *Consider the following specification of the natural numbers:*

$$\begin{array}{ll}\mathsf{SPEC} & NAT\ = \\ [\mathsf{SORT}] & \\ & nat \\ [\mathsf{OP}] & \\ & \mathsf{s} : nat \longrightarrow nat \\ [\mathsf{AX}] & \\ & (1)\ \mathsf{s}(x) \approx \mathsf{s}(y)\ \rightarrow\ x \approx y\end{array}$$

Consider now an alternative specification which introduces an equality test predicate eq *axiomatised with the congruence property:*

$$\begin{array}{ll}\mathsf{SPEC} & NATEQ\ =\ \mathsf{enrich}\ BOOL\ \mathsf{by} \\ [\mathsf{SORT}] & \\ & nat \\ [\mathsf{OP}] & \\ & \mathsf{s} : nat \longrightarrow nat \\ & \mathsf{eq} : nat \times nat \longrightarrow bool \\ [\mathsf{AX}] & \\ & (1)\ \mathsf{eq}(x,x) \approx \mathsf{tt} \\ & (2)\ \mathsf{eq}(x,y) \approx \mathsf{tt}\ \rightarrow\ \mathsf{eq}(y,x) \approx \mathsf{tt} \\ & (3)\ \mathsf{eq}(x,y) \approx \mathsf{tt}\ \wedge\ \mathsf{eq}(y,z) \approx \mathsf{tt}\ \rightarrow\ \mathsf{eq}(x,z) \approx \mathsf{tt} \\ & (4)\ \mathsf{eq}(x,y) \approx \mathsf{tt}\ \rightarrow\ \mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) \approx \mathsf{tt} \\ & (5)\ \mathsf{eq}(\mathsf{s}(x),\mathsf{s}(y)) \approx \mathsf{tt}\ \rightarrow\ \mathsf{eq}(x,y) \approx \mathsf{tt}\end{array}$$

Consider multi-function τ such that

$$\tau(x : nat \approx y : nat) = \{\mathsf{eq}(x : nat, y : nat) \approx \mathsf{tt}\}$$

$NATEQ$ interprets NAT by τ. First note that for any equation $t \approx t'$ such that $NAT \models t \approx t'$, we have that $NATEQ \models \mathsf{eq}(t,t') \approx \mathsf{tt}$, since the interpretation of the proof of $NAT \models t \approx t'$ (in the sense of proposition 1) is a proof of $NATEQ \models \mathsf{eq}(t,t') \approx \mathsf{tt}$. The converse may be verified by induction on the length of the proof of $NATEQ \models \mathsf{eq}(t,t') \approx \mathsf{tt}$.

The example shows how a specification of the natural numbers is interpreted into another one axiomatised exclusively by equations of sort *bool*. Sort nat became hidden, or encapsulated, after refinement.

Other useful design transformations in algebraic specification can similarly be captured as refinements. Our last example illustrates one of them relating specifications to implementations in which some operations are *decomposed* or *mapped to transactions*, *i.e.*, sequences of operations to be executed atomically.

Example 4 *Consider the following fragment of a specification of a bank account management system (BaMS), involving account deposits (operation* deposit*), withdrawals (*withdraw*) and a balance query (*bal*). Assume variables $s : BaMS, i : AccountId$ and $n : \mathbb{N}$, where $\mathbb{N}$ stands for the natural numbers, with the usual arithmetic operators.*

$$\begin{array}{l}\mathsf{SPEC}\ B_1\ = \\ [\mathsf{AX}] \\ \quad (1)\ \mathsf{bal}(\mathsf{deposit}(s,i,n))\ \approx\ \mathsf{bal}(s,i) + n \\ \quad (2)\ \mathsf{bal}(\mathsf{withdraw}(s,i,n))\ \approx\ \mathsf{max}(\mathsf{bal}(s,i) - n, 0) \\ \quad \ldots\ldots\end{array}$$

Consider, now, an implementation B_2, where all transactions are previously validated (through predicate val*). Its axioms include,*

$$\begin{array}{l}(3)\ \mathsf{val}(\mathsf{bal}(\mathsf{val}(\mathsf{deposit}(s,i,n))))\ \approx\ \mathsf{val}(\mathsf{bal}(s,i)) + n \\ (4)\ \mathsf{val}(\mathsf{bal}(\mathsf{val}(\mathsf{withdraw}(s,i,n))))\ \approx\ \mathsf{max}(\mathsf{val}(\mathsf{bal}(s,i)) - n, 0)\end{array}$$

Interpretation

$$\begin{array}{l}\tau_1 : Eq(\Sigma_1) \longrightarrow Eq(\Sigma_2) \\ \quad =\ \{(\mathsf{op}(x) \approx y) \mapsto \{\mathsf{val}(\mathsf{op}(x)) \approx y\} |\ \mathsf{op} \in \Sigma_1\}\end{array}$$

witnesses a refinement in which isolated calls to the operations are mapped to transactions which necessarily include a validation step. The situation where only a given subset of such operations (for example the ones directly affecting an account balance) requires validation is captured by the following axioms in another implementation B_3:

$$\begin{array}{l}(5)\ \mathsf{bal}(\mathsf{val}(\mathsf{deposit}(s,i,n)))\ \approx\ \mathsf{bal}(s,i) + n \\ (6)\ \mathsf{bal}(\mathsf{val}(\mathsf{withdraw}(s,i,n)))\ \approx\ \mathsf{max}(\mathsf{bal}(s,i) - n, 0)\end{array}$$

Clearly,

$$\begin{array}{l}\tau_2 : Eq(\Sigma_1) \longrightarrow Eq(\Sigma_3)\ = \\ \quad \{(\mathsf{op}(x) \approx y) \mapsto \{\mathsf{val}(\mathsf{op}(x)) \approx y\} |\ \mathsf{op} \in \mathsf{userOp}\}\end{array}$$

where $\mathsf{userOp} = \{\mathsf{deposit}, \mathsf{withdraw}\}$*, is the required interpretation. Notice, however, that multi-function*

$$\begin{array}{l}\tau_2' : Eq(\Sigma_1) \longrightarrow Eq(\Sigma_3)\ = \\ \quad \{(\mathsf{op}(x) \approx y) \mapsto \{\mathsf{val}(\mathsf{op}(x)) \approx y, \mathsf{op}(x) \approx y\} |\ \mathsf{op} \in \Sigma_1\}\end{array}$$

is not an interpretation: most of the possible translations of (1) *are not valid in B_3.*

Finally, the reader is invited to check that B_4 is a refinement of B_1 through interpretation τ_3:

$$\tau_3 : Eq(\Sigma_1) \longrightarrow Eq(\Sigma_4) \quad =$$
$$\{(\mathsf{op}(x) \approx y) \mapsto \{\mathsf{val}(\mathsf{op}(x)) \approx y\} | \ \mathsf{op} \in \Sigma_1 \backslash \{\mathsf{withdraw}\}\}$$
$$\cup \{(\mathsf{withdraw}(x) \approx y) \mapsto \{\mathsf{conf}(\mathsf{val}(\mathsf{withdraw}(x))) \approx y\}\}$$

where specification B_4 forces all operations to be validated, but also requires for some of them a previous authorisation, therefore decomposing such operations in three step transactions:

$$(7)\ \mathsf{val}(\mathsf{bal}(\mathsf{val}(\mathsf{deposit}(s,i,n)))) \ \approx \ \mathsf{val}(\mathsf{bal}(s,i)) + n$$
$$(8)\ \mathsf{val}(\mathsf{bal}(\mathsf{conf}(\mathsf{val}(\mathsf{withdraw}(s,i,n)))))$$
$$\approx \ \mathsf{max}(\mathsf{val}(\mathsf{bal}(s,i)) - n, 0)$$

3.4. Stepwise refinement revisited

Having illustrated some typical applications of the notion of refinement put forward in this paper, it is legitimate to ask now how does it relate to the traditional one, based on signature morphisms.

Theorem 5 *Let SP be a specifications over Σ and τ a translation from Σ to Σ' w.r.t. the set of variables X and X' which interprets SP. Then, for every SP' specification over Σ', if $SP^\tau \rightsquigarrow SP'$ then $SP \rightharpoonup_\tau SP'$.*

Proof. Suppose $SP^\tau \rightsquigarrow SP'$, i.e., that $[\![SP']\!] \subseteq [\![SP^\tau]\!]$. Thus, any algebra $A' \in [\![SP']\!]$ is a τ-model of SP. Therefore for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi$ implies $SP' \models \tau(\xi)$. I.e., $SP \rightharpoonup_\tau SP$. □

Theorem 6 *Let $SP = \langle \Sigma, \Phi \rangle$ be a X-flat specification and τ a translation from Σ to Σ' w.r.t. the set of variables X and X'. If τ interprets SP, then the following conditions are equivalent:*

$$SP \rightharpoonup_\tau SP' \quad (1)$$
$$SP^\tau \rightsquigarrow SP' \quad (2)$$

Proof. Theorem 3 justifies implication (1) to (2). The converse implication is just Theorem 5. □

As a corollary we have:

Corollary 1 *Let $SP = \langle \Sigma, \Phi \rangle$ be a X-flat specification and τ a translation from Σ to Σ' w.r.t. the set of variables X and X' which interprets SP. Then, $SP' \models \tau(\Phi)$ implies $SP \rightharpoonup_\tau SP'$.*

We may now try the following question: *given that any mapping f can be regarded as a multifunction, when does a traditional refinement via signature morphism become a refinement via interpretation?* We start with the following lemma:

Lemma 2 *Let SP be a specification over Σ and $\sigma : \Sigma \to \Sigma'$ be an injective signature morphism. Let τ be the translation induced by the signature morphism σ. Then τ interprets SP.*

Proof. Let SP' be the specification over Σ' such that $[\![SP']\!] = \{A' | A \restriction_\sigma \in [\![SP]\!]\}$. Suppose $SP \models \xi$. Let $A' \in [\![SP']\!]$. Since $A' \restriction_\sigma \in [\![SP]\!]$ we have that $A' \restriction_\sigma \models \xi$ and, by Lemma 1, $A' \models \sigma(\xi)$. Therefore $SP' \models \tau(\xi)$. Suppose now $SP' \models \tau(\xi)$ and let $A \in [\![SP]\!]$. Since σ is injective there is $B \in [\![SP']\!]$ such $B \restriction_\sigma = A$. Thus $B \models \tau(\xi)$ and, by Lemma 1, $B \restriction_\sigma \models \xi$, i.e., $A \models \xi$. Hence $SP \models \xi$. Therefore σ interprets SP. □

Next theorem shows that our notion actually is a generalization of the standard one in the special case when the signature morphism is injective.

Theorem 7 *Let SP and SP' be two specifications over Σ and Σ' respectively, and $\sigma : \Sigma \to \Sigma'$ an injective signature morphism. Let τ be the translation induced by the signature morphism σ. Then, $SP \rightsquigarrow_\sigma SP'$ implies $SP \rightharpoonup_\tau SP'$.*

Proof. By previous theorem τ interprets SP. Suppose $SP \rightsquigarrow_\sigma SP'$, i.e., $[\![SP']\!] \restriction_\sigma \subseteq [\![SP]\!]$. Let $\xi \in \mathrm{Ceq}_\Sigma(X)$ such that $SP \models \xi$. Let $A' \in [\![SP']\!]$. Then $A' \restriction_\sigma \in [\![SP]\!]$ and so $A' \restriction_\sigma \models \xi$. By Lemma 1, $A' \models \sigma(\xi)$. Hence $SP' \models \sigma(\xi)$. Therefore $SP \rightharpoonup_\tau SP'$. □

Now, we show that, in the flat case, the two concepts of refinement coincide:

Theorem 8 *Let $\sigma : \Sigma \to \Sigma'$ be an injective signature morphism, $SP = \langle \Sigma, \Phi \rangle$ a X-flat specification and SP a specification over Σ'. Let τ be the translation induced by the signature morphism σ. Then $SP \rightsquigarrow_\sigma SP'$ iff $SP \rightharpoonup_\tau SP'$.*

Proof. Suppose $SP \rightharpoonup_\tau SP'$. By Theorem 5, $SP^\tau \rightsquigarrow SP'$. On the other hand, since $[\![SP^\tau]\!] \subseteq [\![\langle \Sigma', \sigma(\Phi) \rangle]\!]$, by Theorem 1, $SP \rightsquigarrow_\sigma SP^\tau$. Since we can vertically compose σ- refinements we have $SP \rightsquigarrow_\sigma SP'$. □

The discussion concerning the composition of refinements via interpretation is not straightforward. For *vertical composition* an additional property has to be imposed on the components' interpretations. Formally,

Theorem 9 *Let SP, SP' and SP'' be three specifications over Σ, Σ' and Σ'' respectively. Let τ be a translation from Σ to Σ' w.r.t. the set of variables X and X' and ρ a translation from Σ' to Σ'' w.r.t. the set of variables X' and X''. Suppose that $SP \rightharpoonup_\tau SP'$, $SP' \rightharpoonup_\rho SP''$ and ρ interprets SP^τ. Then $SP \rightharpoonup_{\rho \circ \tau} SP''$*

Proof. Let $\xi \in \mathrm{Ceq}_\Sigma(X)$. Suppose $SP \models \xi$. Since $SP \rightharpoonup_\tau SP'$, $SP' \models \tau(\xi)$. And, from $SP' \rightharpoonup_\rho SP''$ we have $SP'' \models \rho(\tau(\xi))$. By hypothesis, τ interprets SP and ρ interprets SP^τ. In particular: SP^τ is a τ-interpretation of SP and $(SP^\tau)^\rho$ is a ρ-interpretation of SP^τ. Hence, for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi \Leftrightarrow SP^\tau \models \tau(\xi)$ and for any $\zeta \in \mathrm{Ceq}_\Sigma(X')$, $SP^\tau \models \zeta \Leftrightarrow (SP^\tau)^\rho \models \tau(\zeta)$. Therefore, for any $\xi \in \mathrm{Ceq}_\Sigma(X)$, $SP \models \xi \Leftrightarrow (SP^\tau)^\rho \models \rho(\tau(\xi))$. That is, $\rho \circ \tau$ interprets SP. □

On the other hand, *horizontal composition* of refinements via interpretations is still a topic of current research. To illustrate the kind of results we are investigating suppose, for example, that τ interprets SP in SP'. The challenge is to prove that τ also interprets any axiomatic extension of SP in an appropriate subspecification of SP' .

4. Conclusions and further work

The paper introduced a new notion of refinement and started to setting the way towards the development of a consistent algebraic theory of refinements via interpretations. The results, characterizations and applications obtained are promising, in the sense that a number of useful transformations of specifications become captured as refinement steps. Although the paper raises more questions than it gives answers, the path seems to be clear.

The main focus of our current work is the integration of the refinements via interpretation within the standard refinement process of algebraic specifications. A first step in this direction concerns the study of hybrid notions like $SP \rightharpoonup_{\tau \triangleright \sigma} SP'$ iff $SP^{\tau} \rightsquigarrow_{\sigma} SP'$, for τ a translation and σ a signature morphism. In this context, our $\tau \triangleright \sigma$-models would consist of algebras of the class $[\![SP^{\sigma}]\!] \restriction_{\sigma}$.

Another topic to explore is the equivalence of algebraic specification up to logical interpretations. As a starting point, it would be worth to explore relation $\equiv$ defined as follows: $SP \equiv SP'$ if there are interpretations τ and ρ such that $SP \rightharpoonup_{\tau} SP'$ and $SP' \rightharpoonup_{\rho} SP$. It is not difficult to see that $SP \models \xi$ implies $SP \models \rho(\tau(\xi))$ and $SP' \models \eta$ implies $SP' \models \tau(\rho(\eta))$. More challenging seems to be a stronger equivalence, studied in the context of equivalence between logical systems [CG05, BP89], which requires interpretations to be *mutually inverse*, that is inverses of one another.

Acknowledgements. This work is funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - the Portuguese Foundation for Science and Technology, within project FCOMP-01-0124-FEDER-007254, as well as under contract PTDC/MAT/68723/2006 and the Unidade de Investigação Matemática e Aplicações of University of Aveiro.

References

[BH06] M. Bidoit and R. Hennicker. Proving behavioral refinements of col-specifications. In *Essays Dedicated to Joseph A. Goguen*, pages 333–354, 2006.

[BP] W. Blok and D. Pigozzi. Abstract algebraic logic and the deduction theorem. Preprint. To appear in the Bulletin of Symbolic Logic. Available at http://www.math.iastate.edu/dpigozzi/papers/aaldedth.pdf.

[BP89] W. Blok and D. Pigozzi. Algebraizable logics. *Memoirs of the American Mathematical Society*, 396, Amer. Math. Soc., Providence, 1989.

[BR03] W. Blok and J. Rebagliato. Algebraic semantics for deductive systems. *Studia Logica*, 74(1-2):153–180, 2003.

[BS81] S. Burris and H. P. Sankappanavar. *A course in universal algebra.* Graduate Texts in Mathematics, Vol. 78. New York - Heidelberg Berlin: Springer-Verlag., 1981.

[BSR04] Don Batory, J. N. Sarvela, and A. Rauschmayer. Scaling step-wise refinement. *IEEE Trans. in Sofware Engineering*, 30(6):355–371, 2004.

[CG05] C. Caleiro and R. Gonçalves. Equipollent logical systems. In *Logica Universalis*, pages 99–111. Birkhäuser, Basel, 2005.

[Cze01] J. Czelakowski. *Protoalgebraic Logics*. Trends in logic, Studia Logica Library, Kluwer Academic Publishers, 2001.

[DD05] Bastian Dolle and Walter Dosch. Transforming functional signatures of algebraic specifications into object-oriented class signatures. In *APSEC '05: Proceedings of the 12th Asia-Pacific Software Engineering Conference*, pages 323–332, Washington, DC, USA, 2005. IEEE Computer Society.

[Fav98] Liliana Favre. Object oriented reuse through algebraic specifications. In *TOOLS '98: Proceedings of the Technology of Object-Oriented Languages and Systems*, page 101, Washington, DC, USA, 1998. IEEE Computer Society.

[FD01] H. A. Feitosa and I. M. L. D'Ottaviano. Conservative translations. *Ann. Pure Appl. Logic*, 108(1-3):205–227, 2001.

[Fei97] H. Feitosa. *Traduções Conservativas*. PhD thesis, Universidade Federal de Campinas,Instituto de Filosofia e Ciências Humanas, 1997.

[Fia04] J. L. Fiadeiro. Software services: scientific challenge or industrial hype? In K. Araki and Z. Liu, editors, *Proc. First International Colloquim on Theoretical Aspects of Computing (ICTAC'04), Guiyang, China*, pages 1–13. Springer Lect. Notes Comp. Sci. (3407), 2004.

[GB92] J. Goguen and R. Burstall. Institutions: abstract model theory for specification and programming. *J. ACM*, 39(1):95–146, 1992.

[Mar06] M. A. Martins. Behavioral institutions and refinements in generalized hidden logics. *Journal of Universal Computer Science*, 12(8):1020–1049, 2006.

[MDT09] T. Mossakowski, R. Diaconescu, and A. Tarlecki. What is a logic translation ? *Logica Universalis*, 2009.

[MHST03] Till Mossakowski, Anne Haxthausen, Donald Sannella, and Andrzej Tarlecki. CASL: The common algebraic specification language: Semantics and proof theory. *Computing and Informatics*, 22:285–321, 2003.

[MP07] M. A. Martins and D. Pigozzi. Behavioural reasoning for conditional equations. *Mathematical. Structures in Comp. Sci.*, 17(5):1075–1113, 2007.

[MT92] K. Meinke and J. V. Tucker. Universal algebra. In *Handbook of logic in computer science, Vol. 1*, volume 1 of *Handb. Log. Comput. Sci.*, pages 189–411. Oxford Univ. Press, New York, 1992.

[San00] D. Sannella. Algebraic specification and program development by stepwise refinement. (Extended abstract). In *Bossi, Annalisa (ed.), Logic-based program synthesis and transformation. 9th international workshop, LOPSTR '99. Venice, Italy, September 22-24, 1999. Selected papers. Berlin: Springer. Lect. Notes Comput. Sci. 1817*, pages 1–9. 2000.

[ST88] D. Sannella and A. Tarlecki. Towards Formal Development of Programs from Algebraic Specifications: Implementations Revisited. *Acta Informatica*, (25):233–281, 1988.

[ST97] D. Sannella and A. Tarlecki. Essential concepts of algebraic specification and program development. *Formal Asp. Comput.*, 9(3):229–269, 1997.

[STar] D. Sannella and A. Tarlecki. *Foundations of Algebraic Specifications and Formal Program Development*. Cambridge University Press, To appear.

[Tar03] A. Tarlecki. Abstract specification theory: An overwiew. In *Models, Algebras, and Logics of Engineering Software,M. Broy, M. Pizka eds.*, NATO Science Series, Computer and Systems Sciences, VOL 191, pages 43–79. IOS Press, 2003.

[Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science (volume B)*, pages 673–788. Elsevier - MIT Press, 1990.