Logical Methods in Computer Science
Vol. 10(1:1)2014, pp. 1–39
www.lmcs-online.org

Submitted Mar. 12, 2011
Published Jan. 2, 2013

THE ROLE OF LOGICAL INTERPRETATIONS IN PROGRAM DEVELOPMENT

MANUEL A. MARTINS [a], ALEXANDRE MADEIRA [b], AND LUÍS S. BARBOSA [c]

[a] CIDMA - Center for R&D in Mathematics and Applications, Dep. Mathematics, Univ. Aveiro, Aveiro, Portugal
e-mail address: martins@ua.pt

[b] HASLab - INESC TEC, Univ. Minho & Dep. Mathematics, Univ. Aveiro, Portugal
e-mail address: madeira@ua.pt

[c] HASLab - INESC TEC, Univ. Minho, Braga, Portugal
e-mail address: lsb@di.uminho.pt

Abstract. Stepwise refinement of algebraic specifications is a well known formal methodology for program development. However, traditional notions of refinement 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 a new approach to refinement in which signature morphisms are replaced by *logical interpretations* as a means to witness refinements. The approach is first presented in the context of equational logic, and later generalised to deductive systems of arbitrary dimension. This allows, for example, refining sentential into equational specifications and the latter into modal ones.

1. Introduction

1.1. **Context.** 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.

This paper's contribution is placed at a specific corner of the broad landscape of formal methods for software development: that of *algebraic specification* [EM85, Wir90, ST97, AKKB99], a family of methods which, having played a pioneering role, constitutes at present a large and mature body of knowledge and active research.

Such methods have a double origin. On the one hand they can be traced back to early work on data abstraction and modular decomposition of programs [Par72, Hoa72, LZ74, Gut75, GH78]. On the other hand, to research on semantics of program specifications building on results from algebraic logic and model theory. Especially relevant in this respect is the original work of the so-called ADJ group [GTWW77, GTW78] whose initial algebra

2012 ACM CCS: [**Theory of computation**]: Semantics and reasoning—Program reasoning—Program verification / Program specifications; Semantics and reasoning—Program semantics; Formal languages and automata theory.

Key words and phrases: Refinement; algebraic specification; deductive system; logical interpretation.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-10(1:1)2014

semantics was the first, full formal approach to software development put forward. This double origin, temporally located around mid seventies, is not surprising: *compositionality* is both a basic requirement in program development and a major asset in algebraic semantics.

The whole area flourished rapidly from the outset: not only different approaches to semantics (final, observational, loose) emerged, but also the initial tie to many-sorted equational logic was soon extended, first to conditional-equational logic, and later to order-sorted, partial and full first-order among other variants. The emergence of the first effective algebraic specification languages — OBJ [GWM+96] and CLEAR [BG80] — overlaps another major development: the introduction of institutions by J. Goguen and R. Burstall [GB92]. Institution theory, which develops model theory independent of the underlying logical system, made possible to decouple specification methodology from the particularities of whatever semantics one may consider more suitable to a specific problem [Dia08].

Moreover, although for a long time the impact of these methods in industry has been limited, a successful effort has been made in the last 15 years towards convergence on generic frameworks with suitable tool support. The COMPASS and, later, the COFI initiative [San01], which lead to the development of CASL [MHST03], are relevant milestones in this process. Besides CASL, CafeOBJ [DF98] and MAUDE [CDE+07] are currently used in several industrial applications and tool development. Actually, research in such methods, either at a foundational or methodological level, found applications in new, unsuspected areas — for example, in documenting service interfaces [HRD08] , characterising contracts in contract-based programming [BH08] or test generation for software composition [YKZZ08].

1.2. **Motivation.** For the working software engineer, a software component is documented by an *interface*, which provides a language through which it interacts with its environment, and a *specification* of the intended meaning of the services provided. This specification is implemented by a concrete piece of software respecting the specified semantics.

Algebraic specification methods build on the observation that these somehow vague concepts from Software Engineering can be framed rigorously in terms of well-known mathematical notions. Thus, an *interface* corresponds to a *signature*, i.e., a set of names for the relevant types, called *sorts*, and a family of service or operation names, classified by their arity and input-output sorts. A signature generates a formal language, giving a rigorous meaning to what we have called before the component's interaction language. Once fixed a signature, a *specification* describes a *class of models* for that signature, and an *implementation* identifies a specific model within such a class. If functions provide suitable abstractions of the services offered by a software component, this analogy can be made even more concrete by identifying *interfaces* with *algebraic signatures*, (denotations of) *specifications* with *classes of algebras*, and *implementations* with specific *algebras*.

The analogy extends to the entire software development process along which components are *refined* by incrementally adding detail and reducing under-specification. Formally, this is a process of structural transformations witnessed by *signature morphisms*, which map functionally sorts and operations from a signature to another respecting the sort translation of functional types.

In such a context this paper raises and discusses the following question: *can more flexible notions of refinement emerge from replacing signature morphisms by some weaker notion of transformation?*

The quest for *weaker notions of transformation* lead us to a different setting, that of Algebraic Logic [FJP03]. The key conceptual tool is that of a *deductive system*, i.e., a formal

language generated by a *signature*, and a *consequence relation*. Interrelating such systems, through maps connecting logical properties, has been studied from early in the last century. Such maps were called *translations* and investigated as part of an ambitious programme addressing tools to handle the multiplicity of logics. As a result, several intuitive notions of translation are scattered in the literature. Many logicians tailored the notion, for their own purposes, to relate specific logics and to obtain specific results. In general, however, a translation is regarded as a map between sets of formulas of different logics such that the image of a theorem is still a theorem. They were used originally to clarify the relationship between classical and constructive logics.

Our starting point is the observation that specifications describe classes of models and those can be naturally associated to deductive systems. Then, translations that both reflect and preserve consequence relations seem interesting candidates to witness weaker forms of refinement. In this paper we will single out a specific sort of translations based on *multifunctions*, *i.e.*, functions mapping an element to a set of elements. Such translations are called *interpretations* and constitute a central tool in the study of equivalent algebraic semantics (see, *e.g.*, [Wój88, BP89, BP01, BR03, Cze01]). A paradigmatic example is the interpretation of the *classical propositional calculus* into the *equational theory of Boolean algebras* (cf. [BP01, Example 4.1.2]). This paper explores interpretations between the deductive systems corresponding to classes of models of specifications as possible witnesses of refinement steps. The notion seems able to capture a number of transformations which are difficult to deal with in classical terms. Examples include data encapsulation and the decomposition of operations into atomic transactions. It also seems promising in the context of new, emerging computing paradigms which entail the need for more flexible approaches to what counts as a valid transformation along the development process (see, for example, [BSR04]).

1.3. **Contribution.** In this context, the contribution of this paper, which combines and extends previous results by the authors reported in [MMB09a] and [MMB09b], is twofold. On the one hand it puts forward a detailed characterisation of refinement witnessed by interpretations, referred in the sequel as *refinement by interpretation*, and exemplifies its potentialities in a number of small, yet illustrative examples.

On the other hand, it renders the whole approach at a sufficiently abstract setting to be applicable over *arbitrary*, technically *k-dimensional*, deductive systems. The dimension fixes the kind of relationship between terms one is interested in. Dimension 2, for example, encompasses equations, regarded as instances of a binary predicate asserting, for example, term equality, bisimilarity, or observational equivalence. Similarly, a unary predicate asserting the validity of a formula is enough to represent a proposition, leading to 1-dimensional deductive systems. Refinement by interpretation in a general, k-dimensional setting provides a suitable context to deal simultaneously with deductive systems arising from classes of models presented in different logics, for example, as a set of equations, propositions or modal formulas.

1.4. **Scope.** Once stated the paper's contributions, it is important to clearly delimit its scope. First of all it should be stressed that the focus of this paper is not placed on *specifications*, understood as syntactic entities which describe in a structured, modular way classes of models, but rather on the *classes of models* themselves, which constitute their

denotations. Deductive systems, the basic tool in our approach, correspond to such classes. This means that the whole area of *specification structuring* [ST06] is, for the moment, left out. Our approach is not concerned with the fact that specifications describing the relevant classes of models are *flat*, i.e. given by a finite set of sentences, or *structured*, *i.e.*, built by systematic application of a number of operators, such as *union*, *translation* or *hiding*, all of them well characterised in the literature and implemented in a number of computer-supported modelling tools.

This does not deny the fundamental importance of specification structuring. Research on this topic started with the introduction of CLEAR [BG80], by the end of the seventies, and its role cannot be underestimated. Actually, the recursive definition of structured specifications provides basic modular procedures for software composition and architecture. Moreover structuring operations allows one to go beyond the specification power of simple, unstructured specifications [Bor02].

Clearly, the approach proposed in this paper can be tuned to specification refinement in a strict sense. In a recent publication [RMMB11] we showed how *refinement by interpretation* can be lifted to the level of structured specifications with the usual operators mentioned above. We believe, however, that by focussing on *classes of models* this piece of research acquires a broader scope of application and is worth on its own. In particular, it pays off when dealing with requirements that cannot be properly formalised in a specification (for example, the property that a controller has a finite number of states). Note this does not entail any loss of expressivity: for each specification, one may recursively compute its denotation (a signature and a class of models) and work directly with them.

On the other hand, the discussion on which operators should be considered in a specification calculus is still active. For example, very recently, reference [DT11] introduced two new operators for specification composition in order to deal with non-protecting importation modes. This further justifies the relevance of a semantic approach as proposed here.

Another concept to make precise is *refinement*. The word is taken here in the broad sense of a transformation mapping an abstract to a more concrete class of models. As such classes are represented by deductive systems, a refinement will map a deductive system into another, while preserving the consequence relation. This is in line with the usual meaning of refinement: all requirements stated at the original level are still valid after refinement. Moreover, it will be shown in the paper that *refinement by interpretation* between classes of models boils down to the standard notion of refinement as inclusion of classes of models whenever the witnessing interpretation is simply an identity.

Finally, a note on the expressivity of deductive systems. Actually, deductive systems can play a double role representing both logics, on top of which all the specification machinery can be developed, and classes of models as discussed above. Clearly, any institution induces, for each signature, a deductive system through its satisfaction relation, and conversely, deductive systems may be viewed as special cases of institutions as discussed in [Vou03]. For example, a deductive system may represent the class of Boolean algebras; but the latter can also be specified as a theory in a suitable institution. Another example is provided by modal logics which can be regarded as both a deductive system or a theory in the first-order (FOL) institution through the standard translation.

This provides a uniform view of seemingly different settings, enabling us to discuss how essentially the same conceptual tool, that of *logical interpretation*, can be used to interrelate logics and refine classes of models both regarded as deductive systems. Reciprocally,

deductive system can be endowed with an algebraic semantics, as discussed in section 4.3. Note that the quest for such a uniform representation of logics and logical theories pops up in other contexts, namely on the design of logical frameworks. A prime example is provided by the logics-as-theories approach proposed by F. Rabe in [Rab08], resorting to a type theoretical framework, and further developed in the context of the LATIN project [CHK+11].

1.5. **Paper structure.** Section 2 introduces *k-dimensional deductive systems* and their semantics following [BP01]. This paves the way to the formulation of refinement by interpretation in a general setting in sections 4 and 5. Before that, however, in section 3, the approach is instantiated for the case of algebraic specifications over the institution of Horn clause logic. This is a popular framework for algebraic specifications which not only deserves attention on its own, but also provides a simpler setting to build up intuitions. Finally, section 6 concludes and suggests some problems deserving further attention.

2. Preliminaries

Specifications of complex systems resort to different logics, and even to their combination. Consequently a characterisation of refinement by interpretation *needs to be orthogonal to whatever logic is used in specifications. This is achieved through the notion of* k-dimensional deductive systems*, of which the equational case is just an instance for* $k = 2$*. This section reviews such systems and their semantics, following* [BP01]*, to provide the background for the sections to follow.*

2.1. **Deductive systems and translations.** Roughly speaking, a deductive system is a general mathematical tool to reason about formulas in a language generated by a signature. Formally, it is defined as a pair $\mathcal{S} = \langle \Sigma, \vdash \rangle$, where Σ is a signature and $\vdash$ is a substitution-invariant consequence relation between sets of formulas and individual formulas. Clearly, any standard sentential logical system, defined in the usual way by a set of axioms and a set of inference rules (for instance, classical and intuitionist propositional calculus, referred to in the sequel as CPC and IPC, respectively), is a deductive system. First order logic can also be formulated as a deductive system [BP89], which shows how broad the concept is. The formal notion of a deductive system, in this abstract perspective, was originally considered by Łukasiewicz and Tarski [Tar56] and intensely studied, from an algebraic point of view, by many logicians. This gave rise to a new, extremely relevant area of Mathematics, that of *abstract algebraic logic* [FJP03].

Although in some literature on algebraic logic this substitution-invariant consequence relation has been called a *logic* (cf. [Cze01]), we adopt along the paper the designation of *deductive system* used by Blok and Pigozzi [BP89]. This terminology allows us to distinguish this concept from the habitual meaning logic has in Computer Science, typically understood as an abstract framework to express specifications and often abstracted as an *institution* [GB92, Dia08] or a π-institution [FS88].

As mentioned above, translation maps were introduced in the early 20th century as a means to interrelate deductive systems. They were first used to understand the relationship between classical and constructive logics. The well-known Gödel translation of classical logic into intuitionistic logic has inspired disperse works on comparing different logics by means

of translations. Illustrative examples include the works of Kolmogorov [Kol77], Glivenko [Gli29], and Gödel [Göd86] involving classical, intuitionist, and modal logics.

To the best of our knowledge the first general definition of translation between deductive systems is due to Prawitz and Malmnäs [PM68]. More recently, Wójcicki [Wój88] presented a systematic study of translations between logics, focussing on inter-relations between sentential logics. And the quest goes on (cf. [MDT09, CG05, CCD09]). At the turn of the century, Silva, D'Ottaviano and Sette [SDS99] proposed a general definition of translation between logics as maps preserving consequence relations. Then, Feitosa and D'Ottaviano studied intensively the subclass of translations that preserve and reflect consequence relations and coined the name *conservative translation* [FD01].

Conservative translations which are able to relate a formula to a set of formulas, and are therefore defined as *multifunctions*, are called *interpretations*. Those which commute with substitutions were originally used in abstract algebraic logic to define a very important class of deductive systems — referred to as *algebraisable* [BP89]. In particular, they abstract the strict relationship between classic propositional logic and the class of Boolean algebras. A deductive system is said to be *algebraisable* whenever there exists a class K of algebras such that the consequence relation induced by K is equivalent to the consequence in the deductive system. Such an equivalence was originally defined by means of two mutually inverse interpretations. Since then, this link between logic and universal algebra has been successfully explored. In particular, for an algebraisable deductive system $\mathcal{S}$, properties of $\mathcal{S}$ can be related to algebraic properties of its equivalent algebraic semantics. This kind of results, of which many examples exist, are often called *bridge theorems*.

2.2. **k-dimensional deductive systems.** In order to broaden the spectrum of application of deductive systems, Blok and Pigozzi et al. [BP01] introduced consequence relations over k-tuples of formulas, for k a non-zero natural number. *k-deductive systems*, the result of this generalisation, are the higher dimensional version of the well known sentential logics. Their theory provides a unified treatment for several deductive systems such as the ones corresponding to assertional, equational, and inequational logics. This generalisation also allows to regard interpretations witnessing algebraisability as a special kind of translations between k-deductive systems.

An *equation*, represented in this paper by a formal expression $t \approx t'$, can be regarded as a pair of terms (or formulas) $\langle t, t' \rangle$. This, in turn, is an instance of a binary predicate standing for the equality of two terms. Similarly, a *unary predicate* asserting the validity of a formula is enough to represent a proposition. The first observation leads to what will be characterised in the sequel as a 2-dimensional deductive system, of which the equational case is a particular instance. The second corresponds, roughly speaking, to sentential logics in a quite broad sense (to include, for example, first-order predicate logic when suitably formalised).

In general, adding a k-ary predicate to a strict universal Horn theory without equality, gives rise to a representation of a k-dimensional deductive system, thus providing a suitable context to deal simultaneously with different specification logics.

We go even a step further considering k-deductive systems over *many sorted* languages, because, in general, software systems manipulate several sorts of data. Almost all notions can be formulated in this broader setting as discussed later. Note there are other generalisations that allow the reuse of arguments and tools from abstract algebraic logic in computer science contexts. Hidden logics, introduced by Pigozzi and Martins in [MP07]

(see also [Mar07] and [BM13]) are a prime example. They have been efficiently used to develop specification and verification methodologies for object oriented software systems. Examples include the Boolean logics, *i.e.*, 1-dimensional multi-sorted logics with Bool as the only visible sort, and equality-test operations for some of the hidden sorts in place of equality predicates.

The syntactic support for k-dimensional deductive systems is that of a k-term. Let $\Sigma = \langle S, \Omega\rangle$ be a signature and $X = (X_s)_{s\in S}$ a S-sorted set of variables. A *k-term* of sort s over signature Σ is a sequence of k Σ-terms, all of the same sort s, $\bar{\varphi}{:}s = \langle \varphi_0{:}s, \ldots, \varphi_{k-1}{:}s\rangle$, abbreviated to $\bar{\varphi}$ whenever references to sorts can be omitted. A *k-variable of sort s* is a sequence of k variables all of the same sort s. $\mathrm{Te}^k_\Sigma(X)$ is the sorted set of all k-terms over Σ with variables in X, i.e.,

$$\mathrm{Te}^k_\Sigma(X) = \langle (\mathrm{Te}_\Sigma(X)_s)^k | s \in S\rangle$$

where $\mathrm{Te}_\Sigma(X)_s$ is the set of all terms over Σ, of sort s, with variables in X. Whenever each $\mathrm{Te}_\Sigma(X)_s$, for each sort s in Σ, is non empty, their union acts as the carrier of the *Σ-term algebra* freely generated from X, which we denote by $\mathrm{Te}_\Sigma(X)$. A *substitution on* $\mathrm{Te}_\Sigma(X)$ is just an endomorphism over $\mathrm{Te}_\Sigma(X)$.

Let us fix some notation and terminology: if $\bar{\varphi}(x_0{:}s_0, \ldots, x_{n-1}{:}s_{n-1})$ is a k-term over Σ, A is a Σ-algebra, and $a_0 \in A_{s_0}, \ldots, a_{n-1} \in A_{s_{n-1}}$, we denote by $\bar{\varphi}^A(a_0, \ldots, a_{n-1})$ the value $\bar{\varphi}$ takes in A when variables $x_0, \ldots, x_{n-1}$ are instantiated respectively by $a_0, \ldots, a_{n-1}$. More precisely, if

$$\bar{\varphi}(x_0, \ldots, x_{n-1}) = \langle \varphi_0(x_0, \ldots, x_{n-1}), \ldots, \varphi_{k-1}(x_0, \ldots, x_{n-1})\rangle,$$

then $\bar{\varphi}^A(a_0, \ldots, a_{n-1}) = h(\bar{\varphi}) := \langle h(\varphi_0), \ldots, h(\varphi_{k-1})\rangle$, where h is any homomorphism from $\mathrm{Te}_\Sigma(X)$ to A such that $h(x_i) = a_i$ for all $i < n$.

Let $\mathrm{VAR} = \langle \mathrm{VAR}_s\rangle_{s\in S}$ be an arbitrary but fixed family of countably infinite disjoint sets VAR_s of variables of sort $s \in S$. Following a typical procedure in similar contexts (*e.g.*, [LEW00]), we assume in the sequel VAR fixed for each set of sorts S and large enough to contain all variables needed. Symbols of variables are obviously disjoint of any other symbol in the signature. As usual in sentential logic frameworks, we will refer to formulas (k-formulas) as synonymous to terms (k-terms respectively). Accordingly, we will denote $\mathrm{Te}_\Sigma(\mathrm{VAR})$ by $\mathrm{Fm}(\Sigma)$. Moreover, for each nonzero natural number k, given a sorted signature Σ, a *k-formula* of sort s over Σ is any element of $(\mathrm{Te}^k_\Sigma(\mathrm{VAR}))_s$. The set of all k-formulas will be denoted by $\mathrm{Fm}^k(\Sigma)$. Also note that an S-sorted subset Γ of k-formulas is identified with the unsorted set $\bigcup_{s\in S}\Gamma_s$, which allows writing $\bar{\varphi} \in \Gamma$ to mean that $\bar{\varphi} \in \Gamma_s$, for some sort s. A set $\Gamma \subseteq \mathrm{Fm}^k(\Sigma)$ is said to be *globally finite* when Γ_s is a finite set for each sort s of Σ, equal to $\emptyset$ except for a finite number of them, *i.e.*, $\bigcup_{s\in S}\Gamma_s$ is finite. In this setting, a k-dimensional deductive system is defined as a substitution-invariant consequence relation on the set of k-formulas. The following definition generalises the one due to W. Blok and D. Pigozzi [BP01] for the one-sorted case.

Definition 2.1. A *k-dimensional deductive system* is a pair $\mathcal{L} = \langle \Sigma, \vdash_\mathcal{L}\rangle$, where Σ is a sorted signature and $\vdash_\mathcal{L} \subseteq \mathcal{P}(\mathrm{Fm}^k(\Sigma)) \times \mathrm{Fm}^k(\Sigma)$ is a relation such that, for all $\Gamma \cup \Delta \cup \{\bar{\gamma}, \bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$, the following conditions hold:

(i) $\Gamma \vdash_\mathcal{L} \bar{\gamma}$ for each $\bar{\gamma} \in \Gamma$;
(ii) if $\Gamma \vdash_\mathcal{L} \bar{\varphi}$, and $\Delta \vdash_\mathcal{L} \bar{\gamma}$ for each $\bar{\gamma} \in \Gamma$, then $\Delta \vdash_\mathcal{L} \bar{\varphi}$;
(iii) if $\Gamma \vdash_\mathcal{L} \bar{\varphi}$, then $\sigma(\Gamma) \vdash_\mathcal{L} \sigma(\bar{\varphi})$ for every substitution σ.

A k-deductive system is *specifiable* if $\vdash_{\mathcal{L}}$ is *compact* (or *finitary* in the terminology of abstract algebraic logic), *i.e.*, if, whenever $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$, there exists a globally finite subset Δ of Γ such that $\Delta \vdash_{\mathcal{L}} \bar{\varphi}$. The relation $\vdash_{\mathcal{L}}$, abbreviated to $\vdash$ whenever $\mathcal{L}$ is clear from the context, is called *the consequence relation of* $\mathcal{L}$.

It is easy to see that, for any $\Gamma \cup \Delta \cup \{\bar{\gamma}, \bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$, $\Gamma \vdash \bar{\gamma}$ and $\Gamma \subseteq \Delta$ imply $\Delta \vdash \bar{\gamma}$.

Every consequence relation $\vdash$ has a natural extension to a relation between sets of k-formulas, also denoted by $\vdash$, defined by $\Gamma \vdash \Delta$ if $\Gamma \vdash \bar{\varphi}$ for each $\bar{\varphi} \in \Delta$. Finally, the relation of *interderivability* between sorted sets is defined by $\Gamma \dashv\vdash \Delta$ if $\Gamma \vdash \Delta$ and $\Delta \vdash \Gamma$. We abbreviate $\Gamma \cup \{\bar{\varphi}_0, \ldots, \bar{\varphi}_{n-1}\} \vdash \bar{\varphi}$ and $\Gamma_0 \cup \cdots \cup \Gamma_{n-1} \vdash \bar{\varphi}$ by $\Gamma, \bar{\varphi}_0, \ldots, \bar{\varphi}_{n-1} \vdash \bar{\varphi}$ and $\Gamma_0, \ldots, \Gamma_{n-1} \vdash \bar{\varphi}$, respectively.

Let $\mathcal{L}$ be a (not necessarily specifiable) k-deductive system. A *thm* of $\mathcal{L}$ is a k-formula $\bar{\varphi}$ such that $\vdash_{\mathcal{L}} \bar{\varphi}$, *i.e.*, $\emptyset \vdash_{\mathcal{L}} \bar{\varphi}$. The set of all theorems is denoted by $\mathrm{Thm}(\mathcal{L})$. An *inference rule* is a pair $\langle \Gamma, \bar{\varphi} \rangle$ where $\Gamma = \{\bar{\varphi}_0, \ldots, \bar{\varphi}_{n-1}\}$ a globally finite set of k-formulas and $\bar{\varphi}$ a k-formula, usually represented as

$$\frac{\bar{\varphi}_0, \ldots, \bar{\varphi}_{n-1}}{\bar{\varphi}_n} \tag{2.1}$$

A rule such as (2.1) is said to be a *derivable rule* of $\mathcal{L}$ if $\{\bar{\varphi}_0, \ldots, \bar{\varphi}_{n-1}\} \vdash_{\mathcal{L}} \bar{\varphi}_n$. A set of k-formulas T closed under the consequence relation, *i.e.*, such that $T \vdash_{\mathcal{L}} \bar{\varphi}$ implies $\bar{\varphi} \in T$, is called a *theory* of $\mathcal{L}$. The set of all theories is denoted by $\mathrm{Th}(\mathcal{L})$; it forms a complete lattice under set-theoretic inclusion, which is algebraic if $\mathcal{L}$ is specifiable. Given any set of k-formulas Γ, the set of all consequences of Γ, in symbols $\mathrm{Cn}_{\mathcal{L}}(\Gamma)$, is the smallest theory that contains Γ. It is easy to see that $\mathrm{Cn}_{\mathcal{L}}(\Gamma) = \{\, \bar{\varphi} \in \mathrm{Fm}^k(\Sigma) : \Gamma \vdash_{\mathcal{L}} \bar{\varphi}\}$. Often, a specifiable k-deductive system is presented in the so-called Hilbert style, *i.e.*, by a set of axioms (k-formulas) and inference rules. We say that a k-formula $\bar{\psi}$ is *directly derivable* from a set Γ of k-formulas by a rule such as (2.1) if there is a substitution $h : \mathrm{Fm}(\Sigma) \to \mathrm{Fm}(\Sigma)$ such that $h(\bar{\varphi}_n) = \bar{\psi}$ and $h(\bar{\varphi}_0), \ldots, h(\bar{\varphi}_{n-1}) \in \Gamma$.

Given a set AX of k-formulas and a set IR of inference rules, we say that $\bar{\psi}$ is *derivable* from Γ by AX and IR, in symbols $\Gamma \vdash_{\mathrm{AX,IR}} \bar{\psi}$, if there is a *proof*, *i.e.*, a finite sequence of k-formulas, $\bar{\psi}_0, \ldots, \bar{\psi}_{n-1}$ such that $\bar{\psi}_{n-1} = \bar{\psi}$, and for each $i < n$, one of the following conditions holds:

(i) $\bar{\psi}_i \in \Gamma$,
(ii) $\bar{\psi}_i$ is a substitution instance of a k-formula in AX,
(iii) $\bar{\psi}_i$ is directly derivable from $\{\bar{\psi}_j : j < i\}$ by one of the inference rules in IR.

It is clear that $\langle \Sigma, \vdash_{\mathrm{AX,IR}} \rangle$ is a specifiable k-deductive system. Moreover, a k-deductive system $\mathcal{L}$ is specifiable iff there exist possibly infinite sets AX and IR, of axioms and inference rules respectively, such that, for any k-formulas $\bar{\psi}$ and any set Γ of k-formulas, $\Gamma \vdash_{\mathcal{L}} \bar{\psi}$ iff $\Gamma \vdash_{\mathrm{AX,IR}} \bar{\psi}$ (see [Cze01] for the one sorted case). This justifies that all the examples of specifiable deductive systems introduced in this paper are presented by a set of axioms and inference rules.

If a deductive system $\mathcal{L}$ is equal to $\langle \Sigma, \vdash_{\mathrm{AX,IR}} \rangle$, for some sets AX and IR with $|\mathrm{AX} \cup \mathrm{IR}| < \omega$, $\mathcal{L}$ is said to be *finitely axiomatisable*. A k-deductive system $\mathcal{L}' = \langle \Sigma, \vdash_{\mathcal{L}'} \rangle$ is an *extension* of the k-deductive system $\mathcal{L} = \langle \Sigma, \vdash_{\mathcal{L}} \rangle$ if $\Gamma \vdash_{\mathcal{L}'} \bar{\varphi}$ whenever $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ for all $\Gamma \cup \{\varphi\} \subseteq \mathrm{Fm}^k(\Sigma)$ (*i.e.*, $\vdash_{\mathcal{L}} \subseteq \vdash_{\mathcal{L}'}$). A k-deductive system $\mathcal{L}'$ is *an extension by axioms and rules of* a specifiable k-deductive system $\mathcal{L}$ if it can be axiomatised by adding axioms and inference rules to the axioms and rules of some axiomatisation of $\mathcal{L}$.

2.3. **The equational case.** Typical examples of k-deductive systems are the ones induced by algebraic specifications (see Section 3). Given a signature Σ, they are defined over pairs of Σ-terms $\langle t, t'\rangle$, representing equations $t \approx t'$, and have therefore dimension $k = 2$. Similarly, Σ-conditional equations can be taken as pairs $\langle \Gamma, e\rangle$ where Γ is a globally finite subset of $\mathrm{Fm}^2(\Sigma)$ and $e \in \mathrm{Fm}^2(\Sigma)$. As a particular case, an equation $t \approx t'$ is a conditional equation without premisses, $\langle \emptyset, t \approx t'\rangle$. In general, a conditional equation $\langle \{t_1 \approx t'_1, \ldots, t_n \approx t'_n\}, t \approx t'\rangle$ will be written as $t_1 \approx t'_1 \wedge \cdots \wedge t_n \approx t'_n \to t \approx t'$. In the sequel we will often use $\mathrm{Eq}(\Sigma)$, instead of $\mathrm{Fm}^2(\Sigma)$, for the set of all equations over VAR, and, similarly, $\mathrm{Ceq}(\Sigma)$, for the set of all Σ-conditional equations over VAR.

Let $\Gamma \cup \{t \approx t'\} \subseteq \mathrm{Fm}^2(\Sigma)$ and A be a Σ-algebra. We write $\Gamma \models_A t \approx t'$ if, for every homomorphism $h : \mathrm{Fm}(\Sigma) \to A$,

$$h(\xi) = h(\eta), \text{ for every } \xi \approx \eta \in \Gamma, \text{ implies } h(t) = h(t').$$

For $\Gamma = \emptyset$, $\emptyset \models_A t \approx t'$ is abbreviated to $\models_A t \approx t'$. An equation $t \approx t'$ is an *identity* of A if $\models_A t \approx t'$. Similarly, a conditional equation $\xi = t_1 \approx t'_1, \ldots, t_n \approx t'_n \to t \approx t'$ is a *quasi-identity* of A if $\{t_1 \approx t'_1, \ldots, t_n \approx t'_n\} \models_A t \approx t'$, which is simply written as $A \models \xi$ when clear from the context.

These definitions extend, as expected, to classes of algebras. Given a class of Σ-algebras K, the (*semantic*) *equational consequence relation* $\models_K$ determined by K is defined by

$$\Gamma \models_K t \approx t' \text{ iff, for every } A \in K,\ \Gamma \models_A t \approx t'.$$

In this case $t \approx t'$ is said to be a K-*consequence* of Γ. When clear from the context, we simply write $K \models \xi$, where $\xi = t_1 \approx t'_1, \ldots, t_n \approx t'_n \to t \approx t'$, for $\{t_1 \approx t'_1, \ldots, t_n \approx t'_n\} \models_K t \approx t'$. Both $\models_A$ and $\models_K$ extend to sets of equations C: $\Gamma \models_A C$ iff $\Gamma \models_A \xi$ for each $\xi \in C$, and respectively for $\models_K$. For a set Φ of quasi-equations, adopting the notational convention explained above and rather standard in Universal Algebra, $A \models \Phi$ stands for $A \models \xi$ for each $\xi \in \Phi$ (analogously for a class K of Σ-algebras).

The equational consequence relation $\models_K$ satisfies the conditions of Definition 2.1. Hence it constitutes an example of a 2-deductive system (perhaps the most important one!) often simply denoted by K.

A class K of Σ-algebras is axiomatised by a set Φ of conditional equations if

$$K = \Big\{A \mid \{t_1 \approx t'_1, \ldots, t_n \approx t'_n\} \models_A t \approx t' \quad \text{for all } t_1 \approx t'_1, \ldots, t_n \approx t'_n \to t \approx t' \in \Phi\Big\}.$$

It can be proved that, if K is a class of Σ-algebras axiomatised by a set Φ of conditional equations, then the relation $\models_K$ is specifiable (see [BR03] for the one-sorted case). In this case it can be defined in the Hilbert style, taking the set of Σ-equations in Φ, together with reflexivity, as the set of axioms, and the set of Σ-conditional equations in Φ, along with symmetry, transitivity and congruence rules, as inference rules. Any specifiable equational deductive system over Σ is the natural extension (by axioms and rules) of the (2-dimensional) free deductive system over Σ denoted by EQ_Σ and defined in Figure 1. Note that the consequence relation associated to EQ_Σ coincides with $\models_{\mathrm{Alg}(\Sigma)}$, where $\mathrm{Alg}(\Sigma)$ is the class of all Σ-algebras.

Recall the discussion of the double role played by deductive systems in subsection 1.4. Actually, it is well known that any deductive system can be seen as a π-institution [FS88], and taking care of some extra technicalities, as an institution. However some special finitary ones are, in a more natural way, viewed as theories of institutions, *i.e.*, as sets of sentences equipped with extra features — the deductive apparatus they induce. For instance,

deductive system EQ_Σ
axioms

$$\langle x:s, x:s\rangle$$

inference rules

$$\frac{\langle x:s, y:s\rangle}{\langle y:s, x:s\rangle}$$

$$\frac{\langle x:s, y:s\rangle, \langle y:s, z:s\rangle}{\langle x:s, z:s\rangle}$$

$$\frac{\langle x_0:s_0, y_0:s_0\rangle, \ldots, \langle x_{n-1}:s_{n-1}, y_{n-1}:s_{n-1}\rangle}{\langle O(x_0,\ldots,x_{n-1}), O(y_0,\ldots,y_{n-1})\rangle}$$

(for each sort s, operation symbol $O : s_0, \ldots, s_{n-1} \to s$ in Σ)

Figure 1: The 2-dimensional free deductive system over Σ.

given a quasivariety of algebras, we can define a (2-dimensional) deductive system over the set of equations by using the identities and quasidentities that define the quasivariety, together with the axioms of reflexivity and the inference rules of symmetry, transitivity and congruence rules. This deductive system can naturally be seen as a theory of the institution of Horn clause logic.

2.4. **k-structures.** As discussed in [BP01], a semantics for arbitrary k-deductive systems needs to go beyond the usual algebraic structures, resorting to algebras endowed with a set of k-tuples. Formally,

Definition 2.2. A *k-structure* over a signature $\Sigma = (S, \Omega)$ is a pair $\mathcal{A} = \langle A, F\rangle$ where A is a Σ-algebra and F is a sorted set $\langle F_s\rangle_{s\in S}$ such that $F_s \subseteq A_s^k$

In this definition, the sorted set F of designated elements of A, can be regarded as a set of truth values on A: a formula holds if its interpretation is one of these elements. This is why F is called a *filter*: for a deductive system representing the constructive propositional calculus on a Boolean algebra the notion boils down to the familiar, Boolean filter.

Let $\mathcal{A} = \langle A, F\rangle$ be a k-structure. A k-formula $\bar\varphi{:}v$ is said to be a *semantic consequence* in $\mathcal{A}$ of a set of k-formulas Γ, in symbols $\Gamma \models_{\mathcal{A}} \bar\varphi$, if, for every assignment $h : \mathrm{VAR} \to A$, $h(\bar\varphi) \in F_v$ whenever $h(\bar\psi) \in F_w$ for every $\bar\psi{:}w \in \Gamma$, where F_s is the component s of the sorted set F. Notice that the same notation is used for the assignment and its natural extension to formulas. A k-formula $\bar\varphi$ is *valid* in $\mathcal{A}$, and conversely $\mathcal{A}$ is a *model* of $\bar\varphi$, if $\emptyset \models_{\mathcal{A}} \bar\varphi$. A rule such as (2.1) is a *validity*, or a *valid rule*, of $\mathcal{A}$, and conversely $\mathcal{A}$ is a *model* of the rule, if $\{\bar\varphi_0, \ldots, \bar\varphi_{n-1}\} \models_{\mathcal{A}} \bar\varphi_n$. A formula $\bar\varphi$ is a *semantic consequence* of a set of k-formulas Γ for an arbitrary class $\mathcal{M}$ of k-structures over Σ, in symbols $\Gamma \models_{\mathcal{M}} \bar\varphi$, if $\Gamma \models_{\mathcal{A}} \bar\varphi$ for each $\mathcal{A} \in \mathcal{M}$. It can be proved that $\models_{\mathcal{M}}$ is always a k-deductive system, even if not always specifiable.

Similarly, a k-formula or a rule is a *validity of* $\mathcal{M}$ if it is a validity of each member of $\mathcal{M}$. A k-structure $\mathcal{A}$ is a *model* of a k-deductive system $\mathcal{L}$ if $\Gamma \vdash_{\mathcal{L}} \bar\varphi$ always implies $\Gamma \models_{\mathcal{A}} \bar\varphi$,

i.e. if every consequence of $\mathcal{L}$ is a semantic consequence of $\mathcal{A}$. The special models whose underlying algebra is the formula algebra, *i.e.*, models of the form $\langle \mathrm{Fm}^k(\Sigma), T\rangle$, for T a theory, are called *Lindenbaum-Tarski models*. The class of all models of $\mathcal{L}$ is denoted by $\mathrm{Mod}(\mathcal{L})$. If $\mathcal{L}$ is a specifiable k-deductive system, then $\mathcal{A}$ is a model of $\mathcal{L}$ iff every axiom and rule of inference is a validity of $\mathcal{A}$.

In the equational case the semantics based on 2-structures boils down to the traditional algebraic semantics. More precisely, given a quasi-equational class K of Σ-algebras (*i.e.* axiomatised by a set of quasi-equations over Σ), the algebra of any model of the 2-deductive system induced by K with the identity as its filter belongs to K.

A class of k-structures $\mathcal{M}$ is a *k-structure semantics* of $\mathcal{L}$ if $\vdash_{\mathcal{L}} = \models_{\mathcal{M}}$. The class of all models of $\mathcal{L}$ forms a k-structure semantics of $\mathcal{L}$. This fact is expressed in a specific completeness theorem [MP07], stating that, for any k-deductive system $\mathcal{L}$, $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ iff $\Gamma \models_{\mathrm{Mod}(\mathcal{L})} \bar{\varphi}$, for every $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$.

3. Refinement by interpretation: The equational case

This section introduces, exemplifies and discusses the concept of refinement by interpretation *— the core contribution of the paper — framed in the specific, but popular, setting of algebraic specification over the institution of Horn clause logic.*

3.1. **Algebraic specification and refinement.** This section introduces refinement by logical interpretation for the specific case of algebraic specifications. As usual (see *e.g.*, [ST11]), an *algebraic specification* SP is considered here as a structured specification over the institution of Horn clause logic (**HCL**) [Dia08]. Each SP denotes a pair $\langle \Sigma, [\![SP]\!]\rangle$ where Σ is a signature and $[\![SP]\!]$ is a class of Σ-algebras. The class $[\![SP]\!]$ of Σ-algebras is called the *model class of* SP, and each Σ-algebra in $[\![SP]\!]$ a *model of* SP. If ξ is a conditional equation $\langle \Gamma, e\rangle$ (respectively, an equation e), we write $SP \models \xi$ for $\Gamma \models_{[\![SP]\!]} e$ (respectively, $\models_{[\![SP]\!]} e$). Both cases extend, as expected, to sets of conditional equations (respectively, equations).

When an algebraic specification SP is *flat*, or *basic*, its model class $[\![SP]\!]$ of algebras is axiomatised by a set Φ of conditional equations. In this case SP is the pair $\langle \Sigma, \Phi\rangle$. If the definition is restricted to formulas over a specific set of Σ-variables $X \subseteq \mathrm{VAR}$, SP is said *X-flat*. When Φ is a set of equations the flat specification $SP = \langle \Sigma, \Phi\rangle$ is called a (flat) *equational specification*. Recall that a class K of Σ-algebras axiomatisable by a set of equations is called a *variety*. A variety can also be characterised as a nonempty class K of Σ-algebras closed under homomorphic images, subalgebras and direct products (cf. [BS81], [ST11]). This famous result, due to Birkhoff, turns out to be very useful to show that a given algebraic specification is not an equational specification. In the sequel, for simplicity, when clear from the context, we refer to algebraic specifications simply as specifications.

In this context *stepwise refinement* [ST97, Mar06] of specifications refers to the process through which a complex design is produced by incrementally adding details and reducing under-specification. This proceeds step-by-step until the class of models becomes restricted to such an extent that a program can be easily manufactured. Formally, given a specification SP, the implementation process builds a correct realisation from a concrete enough class of Σ-models K such that K is a subset of the class of denotations of SP. During this process,

the specification is enriched according to specific design decisions, iteratively approaching the intended meaning for the final program.

Starting from an initial abstract specification SP_0, refinement builds a chain of specifications

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

where, for $1 \leq i \leq n$, $SP_{i-1} \rightsquigarrow SP_i$ represents reverse inclusion of the respective classes of models. Transitivity of inclusion assures that $SP_0 \rightsquigarrow SP_n$. From SP_1 onwards each element in this chain is the result of a *refinement step*.

In order to deal with more complex requirements along the implementation process, for example to enable the possibility of renaming, adding or grouping together different signature components, refinement steps $SP' \rightsquigarrow SP$ are traditionally taken up to *signature morphisms*. Recall that a *signature morphism* from $\Sigma = \langle S, \Omega \rangle$ to $\Sigma' = \langle S', \Omega' \rangle$ is a pair $\sigma = \langle \sigma_{sort}, \sigma_{op} \rangle$, where $\sigma_{sort} : S \to S'$ and σ_{op} is a $(S^* \times S)$-family of functions respecting the sorts of operation names in Ω, *i.e.*, $\sigma_{op} = (\sigma_{\omega,s} : \Omega_{\omega,s} \to \Omega'_{\sigma^*_{sort}(\omega),\sigma_{sort}(s)})_{\omega \in S^*, s \in S}$ (where for $\omega = s_1 \ldots s_n \in S^*, \sigma^*_{sort}(\omega) = \sigma_{sort}(s_1) \ldots \sigma_{sort}(s_n)$).

Given a signature morphism $\sigma : \Sigma \to \Sigma'$ and a Σ'-algebra A, let $A\restriction_\sigma$ denote the *reduct* of A along σ, *i.e.*, for any $s \in S$, $s^{(A\restriction_\sigma)} = \sigma(s)^A$, and for all $f : s_1, \ldots, s_n \to s \in \Sigma$, $f^{A\restriction_\sigma} = \sigma_{op}(f)^A$. The notation s^A and f^A refer, respectively, to the carrier of sort s and the interpretation of operation symbol f in the algebra A. In this context we say that SP' is a refinement of SP witnessed by σ, or simply a σ-refinement, when $[\![SP']\!]\restriction_\sigma \subseteq [\![SP]\!]$, where $[\![SP']\!]\restriction_\sigma = \{A\restriction_\sigma \mid A \in [\![SP']\!]\}$.

Having fixed the notation and the basic concepts we may now jump to the kernel of this section: representing classes of specification models as 2-deductive systems and introduce interpretations as possible witnesses to the refinement steps.

3.2. **Denotations of algebraic specifications as 2-deductive systems.** As clarified in the Introduction (subsection 1.4), our approach is based on the representation of specifications' model classes as deductive systems. Our focus is essentially semantic, *i.e.*, on what specifications denote, and, therefore, the whole theory of refinement by interpretation discussed here is independent from any specification structuring mechanism.

Actually, any specification SP denotes a class of algebras — its *model class*, $[\![SP]\!]$. This in turn induces a 2-deductive system according to the procedure sketched in subsection 2.3. Its consequence relation is $\models_{[\![SP]\!]}$. The possibility of this consequence relation being non finitary, for example if arising from non flat specifications, is also covered in this approach.

As we will see, it turns out that k-deductive systems are an efficient universal tool to develop, in this way, a theory about all classes of models over a fixed signature. A most important particular case is that of *flat* specifications: each of them can be identified with the deductive system it induces. This observation will be assumed throughout the text and in the examples. Moreover, using interpretations between the induced deductive systems, one may go from one model class to another.

In a sense to become clearer below, we say that a specification SP' *refines* another specification SP *by interpretation* τ, if τ is an interpretation of SP such that $SP \models \xi \Rightarrow SP' \models \tau(\xi)$ for any formula ξ. Before jumping to the technical definition, consider the following example which, although elementary, may help to build up some intuitions for this notion of refinement.

Example 3.1. The introduction of a two-element Boolean algebra together with equality tests for each sort allows the software engineer to formulate arbitrary universal first-order axioms as equations in the Boolean sort. The importance of this construction comes from the fact that, although many of the most natural specification conditions take the form of universal first-order sentences, only equational and conditional-equational axioms are guaranteed to possess an initial model. This motivation, as well as equality-test algebras in general, are extensively discussed in [Pig91].

Consider, thus, two flat specifications S and T. The former has a signature Σ which declares a sort s and a function $f : s \longrightarrow s$. S has an empty set of axioms; thus, the corresponding class of models consists of all algebras over its signature. On the other hand, specification T is depicted in Figure 2.

spec T
sorts
s
ops
$ok :\longrightarrow s$
$f : s \longrightarrow s$
$test : s \times s \longrightarrow s$
axioms
$test(x, x) \approx ok$
$(test(x, x') \approx ok\ ,\ test(x', x'') \approx ok) \ \rightarrow\ test(x, x'') \approx ok$
$test(x, x') \approx ok \ \rightarrow\ test(x', x) \approx ok$
$test(x, x') \approx ok \ \rightarrow\ test(f(x), f(x')) \approx ok$

Figure 2: Specification T.

It is not difficult to see, by induction on the structure of proofs, that the translation

$$\begin{array}{rccc} \tau : & \mathrm{Eq(Sig(S))} & \rightarrow & \mathrm{Eq(Sig(T))} \\ & x \approx x' & \mapsto & test(x, x') \approx ok \end{array}$$

interprets S in the sense that the consequence relation induced by S is preserved and reflected by τ.

An inspection of the signatures of both specifications shows that there exists an unique signature morphism definable between them: the inclusion $\iota : \mathrm{Sig(S)} \rightarrow \mathrm{Sig(T)}$. This morphism induces the identity translation between formulas which, obviously, does not interpret the specification above. $\diamond$

This kind of anomalies can be circumvented by tailoring refinement to the specific situation in hands. In the example above equality may be taken as an extralogical predicate interpreted, roughly speaking, in the second specification as the test operation. However, such solutions just hold for special cases and have to be redefined case by case. The approach proposed in this paper aims at providing a uniform, general solution.

3.3. **Translations.** A number of notions of translation between logical systems have been proposed in the literature (see, for example, [Fei97, FD01, BP01, MDT09]).

Definition 3.2 (Translation)**.** Let $\Sigma = (S, \Omega)$, $\Sigma' = (S', \Omega')$ be two signatures. A *translation from* Σ *to* Σ' is a globally finite sorted multifunction[1] $\tau : \mathrm{Eq}(\Sigma) \prec\!\!\!\triangleleft \mathrm{Eq}(\Sigma')$. More precisely, τ maps each equation in $\mathrm{Eq}(\Sigma)$ into a globally finite S'-sorted set of equations in $\mathrm{Eq}(\Sigma')$.

When $\Sigma = \Sigma'$, τ is called a *self translation of* Σ. In this case, we say that τ *commutes with substitutions* if, for every substitution σ, and every equation $e \in \mathrm{Eq}(\Sigma)$, $\tau(\sigma(e)) = \sigma(\tau(e))$, where, again, the same notation is used to denote the application of a substitution to a formula or a set of formulas.

Any translation $\tau : \mathrm{Eq}(\Sigma) \prec\!\!\!\triangleleft \mathrm{Eq}(\Sigma')$ can be extended to conditional equations as the multifunction $\tau^* : \mathrm{Ceq}(\Sigma) \prec\!\!\!\triangleleft \mathrm{Ceq}(\Sigma')$ given by

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

for $\xi = \langle \Gamma, e \rangle$ a conditional equation. In the sequel, we identify τ^* with τ. The reason for requiring that the image of τ_s, for each sort s, to be globally finite becomes now clear from the definition of τ^*. The following lemma establishes an important result.

Lemma 3.3. *Let $\Sigma = (S, \Omega)$ be a standard signature and τ a self translation of Σ. Then the following conditions are equivalent:*

(i) *τ commutes with substitutions.*
(ii) *There exist variables $x, y \in \mathrm{VAR}$ and an S-sorted set of equations $E(x, y)$ in these two variables such that, for any $t \approx t' \in \mathrm{Eq}(\Sigma)_s$, $\tau_s(t \approx t') = E_s(t, t')$.*

Proof. Assume that τ commutes with arbitrary substitutions, fix distinct variables x, y, and define $E := \tau(x \approx y)$. Let $\mathrm{FV}(E)$ denote the set of free variables occurring in E, and suppose $\mathrm{FV}(E) \subseteq \{x, y, r_1, r_2, \dots\}$. Let e be a substitution such that $e(x) = x$, $e(y) = y$, and $e(r_i) = x$ for all $i \geq 1$. By assumption, $E(x, y, x, \dots) = E(e(x), e(y), e(r_1), ...) = e(E) = e\big(\tau(x \approx y)\big) = \tau\big(e(x) \approx e(y)\big) = \tau(x \approx y) = E(x, y, r_1, r_2, \dots)$. Hence, $\{r_1, r_2, \dots\} \subseteq \{x, y\}$. Thus $\mathrm{FV}(E) \subseteq \{x, y\}$. We will write $E(x, y)$ to denote that the variables which occur in E are among x and y. Now, let $t \approx t' \in \mathrm{Eq}(\Sigma)$, and e be a substitution such that $e(x) = t$ and $e(y) = t'$. We have that $\tau(t \approx t') = \tau\big(e(x) \approx e(y)\big) = e\big(\tau(x \approx y)\big) = e\big(E(x, y)\big) = E\big(e(x), e(y)\big) = E(t, t')$. Suppose now that ((ii)) holds. Let α be a substitution in Σ. Then, for any $t \approx t' \in \mathrm{Eq}(\Sigma)$,

$$\alpha(\tau(t \approx t')) \;=\; \alpha(E_s(t, t')) \;=\; E_s(\alpha(t), \alpha(t'))) \;=\; \tau(\alpha(t) \approx \alpha(t')) \;=\; \tau(\alpha(t \approx t')). \qquad \square$$

3.4. **Interpretations.** Defined as a multifunction, a translation maps a formula into a set of formulas, and this is what makes translations interesting to establish relationships between specifications. Recall that 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. The following definition singles out the relevant ones:

[1] In the sequel, notation $m : A \prec\!\!\!\triangleleft B$ is used for multifunctions m from A to B.

Definition 3.4 (Interpretation)**.** Let τ be a translation from Σ to Σ', and SP a specification over Σ. The translation τ *interprets* SP if there is a class of algebras K over Σ' such that, for any $\xi \in \mathrm{Ceq}(\Sigma)$,

$$SP \models \xi \text{ if and only if } K \models \tau(\xi)$$

In this case we say that τ *interprets* SP *in* K, and K *is a* τ*-interpretation of* SP. Moreover, when K is the denotation of a specification SP' we say that τ *interprets* SP *in* SP', and SP' *is a* τ*-interpretation of* SP.

Example 3.5. The interpretation of the specification of *Boolean algebras* into the class HA of *Heyting algebras* is a classical example of an interpretation. Let Σ be the usual signature of Boolean algebras (and of Heyting algebras). Consider the well known *double negation* (propositional) map: $\iota(t) = \neg\neg t$.
Let τ be a self translation of Σ defined by

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

It can be shown that τ interprets the specification of Boolean algebras in the class HA [BR03]. This is known as Glivenko's interpretation [Gli29]. It establishes a strict relationship between classical and intuitionistic derivability: if φ is a theorem in CPC then $\neg\neg\varphi$ is a theorem in IPC. The result builds a bridge between the algebraic semantics of *classical propositional calculus* and that of *intuitionistic propositional calculus.* ◇

It is not difficult to see that,

Theorem 3.6. *Let SP and SP' be two algebraic specifications over a signature Σ, and τ a recursive self translation of Σ that commutes with arbitrary substitutions and interprets SP in SP'. If SP is decidable then SP' is decidable.*

Proof. In [Fei97]. □

Definition 3.7. Let τ be a translation from Σ to Σ' and SP a specification over Σ. A Σ'-algebra A' is a τ*-model of* SP if, for any $\xi \in \mathrm{Ceq}(\Sigma)$, $SP \models \xi$ implies $A' \models \tau(\xi)$. The τ*-model class of* SP, denoted by $\mathrm{Mod}_\tau(SP)$, is the class of all τ-models of SP.

Observe that, for any conditional equation $\xi = \langle \Gamma, e \rangle$, $SP \models \xi$ implies $\mathrm{Mod}_\tau(SP) \models \tau(\xi)$.

Theorem 3.8. *Let SP be a specification over Σ, and τ a translation from Σ to Σ' that interprets SP. Then the class of models $\mathrm{Mod}_\tau(SP)$ is the largest τ-interpretation of SP.*

Proof. Suppose that τ interprets SP. Let K be a class of models which is a τ-interpretation of SP. Then for any $\xi \in \mathrm{Ceq}(\Sigma)$, $SP \models \xi$ if and only if $K \models \tau(\xi)$. Hence all models in K are τ-models of SP. Thus, $K \subseteq \mathrm{Mod}_\tau(SP)$.

So, we only need to prove that $\mathrm{Mod}_\tau(SP)$ is a τ-interpretation of SP. Let $\xi \in \mathrm{Ceq}(\Sigma)$. It is clear that $SP \models \xi$ implies $\mathrm{Mod}_\tau(SP) \models \tau(\xi)$. Suppose now that $\mathrm{Mod}_\tau(SP) \models \tau(\xi)$. Let K be a class of models that is a τ-interpretation of SP (it exists since τ interprets SP). As proved above, $K \subseteq \mathrm{Mod}_\tau(SP)$, hence $K \models \tau(\xi)$. Finally, K being a τ-interpretation of SP entails $SP \models \xi$. □

The next theorem states that if a specification SP is finitely axiomatisable, so is $\mathrm{Mod}_\tau(SP)$. Therefore there is a flat specification SP^τ such that $[\![SP^\tau]\!] = \mathrm{Mod}_\tau(SP)$.

Theorem 3.9. *Let τ be a self translation of Σ and $SP = \langle \Sigma, \Phi \rangle$ a X-flat specification for a set $X \subseteq \mathrm{VAR}$ of variables. If τ commutes with substitutions then the class of models denoted by the flat specification $SP^\tau = \langle \Sigma, \tau(\Phi) \rangle$ coincides with $\mathrm{Mod}_\tau(SP)$. Moreover, if Φ is finite then SP^τ is finitely axiomatisable.*

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

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 the deductive system $\models_{[\![SP]\!]}$ induced by SP, that $A \models \tau(\xi)$. Therefore A is a τ-model of SP and so, $[\![\langle \Sigma, \tau(\Phi) \rangle]\!] \subseteq \mathrm{Mod}_\tau(SP)$.

Clearly, if Φ is finite then $\tau(\Phi)$ is also finite. Moreover, it can be proved that $\tau(\Phi)$ constitutes an axiomatisation for SP^τ. □

3.5. **Refinement by interpretation.** Logical interpretation, as introduced in the previous section, provides the basic tool for the following definition:

Definition 3.10 (Refinement by interpretation)**.** Let SP be a specification over Σ and τ a translation from Σ to Σ' which interprets SP. We say that a specification SP' over Σ' *refines SP by interpretation τ*, in symbols $SP \rightharpoonup_\tau SP'$, if for any $\xi \in \mathrm{Ceq}(\Sigma)$

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

It is not difficult to see that SP' refines SP by interpretation τ whenever τ interprets SP in $[\![SP']\!]$.

Let us consider some examples of refinement by interpretation. 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 3.11. Consider the specifications of Boolean and Heyting algebras depicted in Figures 3 and 4, where DISTLATTICE is the specification of distributive lattices (see, [BS81]). We assume that a sort *bool* is declared in DISTLATTICE. Note that in this example, as in a few others in the sequel, a naive use is made of a few standard operations for structuring specifications. For example, annotation **enrich** $\mathcal{S}$ means that the (finitary) specification is obtained by adding new operation symbols, new sorts or/and new axioms to $\mathcal{S}$. This is done just for syntactical convenience: to represent an equivalent flat specification, whose existence is trivially shown for both BOOL, HEYTING and all the other cases where we use the same artifice. What should be kept in mind is that the flat specifications correspond directly to deductive systems.
As in Example 3.5, the multifunction τ defined by

$$\tau(p \approx q) \;=\; \{\neg\neg p \approx \neg\neg q\}$$

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

A Gödel algebra, also known as a L-algebra [BD74], is a Heyting algebra that satisfies the pre-linearity condition: $(x \twoheadrightarrow y) \vee (y \twoheadrightarrow x) = \mathsf{tt}$. The class GODEL of all Gödel algebras forms a subvariety of the class HEYTING and is thus a denotation of a flat specification, also denoted by GODEL, obtained from the HEYTING specification by adding the extra

spec BOOL
enrich DISTLATTICE
ops
$\mathsf{tt} : \longrightarrow bool$
$\mathsf{ff} : \longrightarrow bool$
$\neg : bool \longrightarrow bool$
$\wedge : bool \times bool \longrightarrow bool$
$\vee : bool \times bool \longrightarrow bool$
axioms
$p \vee \neg p \approx \mathsf{tt}$
$p \wedge \neg p \approx \mathsf{ff}$
$p \vee \mathsf{tt} \approx \mathsf{tt}$
$p \wedge \mathsf{ff} \approx \mathsf{ff}$

Figure 3: A specification of Boolean algebras.

spec HEYTING
enrich DISTLATTICE
ops
$\mathsf{tt} : \longrightarrow bool$
$\mathsf{ff} : \longrightarrow bool$
$\neg : bool \longrightarrow bool$
$\wedge : bool \times bool \longrightarrow bool$
$\vee : bool \times bool \longrightarrow bool$
$\twoheadrightarrow : bool \times bool \longrightarrow bool$
axioms
$p \vee \mathsf{tt} \approx \mathsf{tt}$
$p \wedge \mathsf{ff} \approx \mathsf{ff}$
$p \twoheadrightarrow p \approx \mathsf{tt}$
$(p \twoheadrightarrow q) \wedge q \approx q$
$p \twoheadrightarrow (q \wedge r) \approx (p \twoheadrightarrow q) \wedge (p \twoheadrightarrow r)$
$p \wedge (p \twoheadrightarrow q) \approx p \wedge q$
$(p \vee q) \twoheadrightarrow r \approx (p \twoheadrightarrow r) \wedge (q \twoheadrightarrow r)$
$\neg p \approx p \twoheadrightarrow \mathsf{ff}$

Figure 4: A specification of Heyting algebras.

axiom $(x \twoheadrightarrow y) \vee (y \twoheadrightarrow x) = \mathsf{tt}$. So GODEL is an example of a refinement by interpretation of the specification of Boolean algebras. $\diamond$

Our next example, although quite elementary, illustrates a key point. It shows how refinement by interpretation may 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

spec NAT
sorts
nat
ops
$\mathsf{z} : \longrightarrow nat$
$\mathsf{s} : nat \longrightarrow nat$
$+ : nat \times nat \longrightarrow nat$
axioms
$x + \mathsf{z} \approx x$
$\mathsf{s}(x + y) \approx x + \mathsf{s}(y)$
$\mathsf{s}(x) \approx \mathsf{s}(y) \rightarrow x \approx y$

Figure 5: A specification of the natural numbers.

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]. In the following example a specification of the natural numbers is interpreted into another one exclusively axiomatised by equations of sort *bool*. Sort *nat* becomes hidden, or encapsulated, after refinement.

Example 3.12. Consider the specification NAT of the natural numbers depicted in Figure 5. An alternative specification, NATEQ, is shown in Figure 6, which introduces an equality test, eq, axiomatised with the congruence property.
NATEQ interprets NAT through multifunction τ defined as

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

First note that for any equation $t \approx t'$ such that NAT $\models t \approx t'$, one also has NATEQ $\models \mathsf{eq}(t, t') \approx \mathsf{tt}$, since the interpretation of the proof of NAT $\models t \approx t'$ is a proof of NATEQ $\models \mathsf{eq}(t, t') \approx \mathsf{tt}$. The converse is proved by induction on the length of the proof of NATEQ $\models \mathsf{eq}(t, t') \approx \mathsf{tt}$. Hence, any refinement of NATEQ, for example, the one obtained by adding axiom $\mathsf{eq}(\mathsf{z}, \mathsf{s}(\mathsf{z})) \approx \mathsf{ff}$, is a refinement by τ of NAT. ◇

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

Example 3.13. 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).

spec BAMS
enrich INT
axioms
$\mathsf{bal}(\mathsf{deposit}(s, i, n), i) \approx \mathsf{bal}(s, i) + n$
$\mathsf{bal}(\mathsf{withdraw}(s, i, n), i) \approx \mathsf{max}(\mathsf{bal}(s, i) - n, 0)$
...

spec NATEQ
enrich BOOL
sorts
nat
ops
$\mathsf{z} : \longrightarrow nat$
$\mathsf{s} : nat \longrightarrow nat$
$+ : nat \times nat \longrightarrow nat$
$\mathsf{eq} : nat \times nat \longrightarrow bool$
axioms
$\mathsf{eq}(x, x) \approx \mathsf{tt}$
$\mathsf{eq}(x, y) \approx \mathsf{tt} \ \rightarrow \ \mathsf{eq}(y, x) \approx \mathsf{tt}$
$\mathsf{eq}(x, y) \approx \mathsf{tt} \ , \ \mathsf{eq}(y, \mathsf{z}) \approx \mathsf{tt} \ \rightarrow \ \mathsf{eq}(x, \mathsf{z}) \approx \mathsf{tt}$
$\mathsf{eq}(x, y) \approx \mathsf{tt} \ \rightarrow \ \mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) \approx \mathsf{tt}$
$\mathsf{eq}(\mathsf{s}(x), \mathsf{s}(y)) \approx \mathsf{tt} \ \rightarrow \ \mathsf{eq}(x, y) \approx \mathsf{tt}$
$\mathsf{eq}(x + \mathsf{z}, x) \approx \mathsf{tt}$
$\mathsf{eq}(\mathsf{s}(x + y), x + \mathsf{s}(y)) \approx \mathsf{tt}$

Figure 6: Hiding sort nat.

Assume INT as the usual flat specification of integer numbers with arithmetic operations, and variables $s{:}Sys$, $i{:}Ac$ and $n, n'{:}Int$, where Sys and Ac are the sorts of bank systems and account identifiers, respectively. The signatures of the main operations are as follows: $\mathsf{deposit} : Sys \times Ac \times Int \longrightarrow Sys$, $\mathsf{withdraw} : Sys \times Ac \times Int \longrightarrow Sys$ and $\mathsf{bal} : Sys \times Ac \longrightarrow Int$.
Consider now an implementation B2 where all debit and credit transactions require a previous validation step. This is achieved through an operation $\mathsf{valid} : Sys \times Ac \times Int \longrightarrow Int$, which, given a bank system state, an account identifier, and a value to be added or subtracted to the account's balance, verifies if the operation can proceed or not. In the first case it outputs the original amount; in the second 0 is returned as an error value. This will force an invalid deposit or withdrawal to have no effect (0 will be added to, or subtracteded from the account's balance). Although this is a quite common form of error recovery (invoking the intended operation with its identity element), note that, for the purpose of this example, the concrete behaviour of operation valid is not relevant as long as an integer is returned. The axioms for B2 include,

spec B2
enrich INT
axioms
...
$\mathsf{bal}(\mathsf{deposit}(s, i, \mathsf{valid}(s, i, n)), i) \approx \mathsf{bal}(s, i) + \mathsf{valid}(s, i, n)$
$\mathsf{bal}(\mathsf{withdraw}(s, i, \mathsf{valid}(s, i, n)), i) \ \approx \ \mathsf{max}(\mathsf{bal}(s, i) - \mathsf{valid}(s, i, n), 0)$

The interpretation $\tau_1 : \mathrm{Eq}(\Sigma_{\mathrm{BAMS}}) \preccurlyeq \mathrm{Eq}(\Sigma_{\mathrm{B2}})$, defined by

$$\tau_1(t \approx t') = \{\gamma \approx \gamma' | \gamma \in \tau_1^{\#}(t) \text{ and } \gamma' \in \tau_1^{\#}(t')\}, \text{ where}$$

$$\tau_1^{\#}(x) \;=\; \{x\} \qquad \text{for } x \in \text{VAR}$$

$$\tau_1^{\#}(f(t_1,t_2,t_3)) \;=\; \{f(t_1',t_2',\mathsf{valid}(t_1',t_2',t_3')) \mid \bigwedge_{i=1..3} t_i' \in \tau_1^{\#}(t_i)\} \quad \text{for } f \in \{\mathsf{deposit},\mathsf{withdraw}\}$$

$$\tau_1^{\#}(f(t_1,\ldots,t_n)) \;=\; \{f(t_1',\ldots,t_n') \mid \bigwedge_{i=1..n} t_i' \in \tau_1^{\#}(t_i)\} \qquad \text{for } f \notin \{\mathsf{deposit},\mathsf{withdraw}\}$$

witnesses a refinement in which isolated calls to the operations are mapped to validated transactions.

It might also be the case that some operations can be executed with or without validation, as in, for example,

spec B3
enrich INT
axioms
...
$\mathsf{bal}(\mathsf{deposit}(s,i,\mathsf{valid}(s,i,n)),i) \;\approx\; \mathsf{bal}(s,i) + \mathsf{valid}(s,i,n)$
$\mathsf{bal}(\mathsf{deposit}(s,i,n),i) \;\approx\; \mathsf{bal}(s,i) + n$
$\mathsf{bal}(\mathsf{withdraw}(s,i,\mathsf{valid}(s,i,n)),i) \;\approx\; \mathsf{max}(\mathsf{bal}(s,i) - \mathsf{valid}(s,i,n), 0)$

Clearly, B3 refines BAMS through the interpretation $\tau_2 : \text{Eq}(\Sigma_{\text{BAMS}}) \preccurlyeq \text{Eq}(\Sigma_{\text{B3}})$,

$$\tau_2^{\#}(x) \;=\; \{x\} \quad \text{for } x \in \text{VAR}$$

$$\tau_2^{\#}(\mathsf{deposit}(t_1,t_2,t_3)) \;=\; \{\mathsf{deposit}(t_1',t_2',\mathsf{valid}(t_1',t_2',t_3')),\mathsf{deposit}(t_1',t_2',t_3') \mid \bigwedge_{i=1..3} t_i' \in \tau_2^{\#}(t_i)\}$$

$$\tau_2^{\#}(\mathsf{withdraw}(t_1,t_2,t_3)) \;=\; \{\mathsf{withdraw}(t_1',t_2',\mathsf{valid}(t_1',t_2',t_3')) \mid \bigwedge_{i=1..3} t_i' \in \tau_2^{\#}(t_i)\}$$

$$\tau_2^{\#}(f(t_1,\ldots,t_n)) \;=\; \{f(t_1',\ldots,t_n') \mid \bigwedge_{i=1..n} t_i' \in \tau_2^{\#}(t_i)\} \text{ for } f \notin \{\mathsf{deposit},\mathsf{withdraw}\}$$

Finally, the reader is invited to check that B4 is a refinement of BAMS through interpretation $\tau_3 : \text{Eq}(\Sigma_{\text{BAMS}}) \to \text{Eq}(\Sigma_{\text{B4}})$, defined similarly to $\tau_2^{\#}$ but for the withdraw case which becomes

$$\tau_3^{\#}(\mathsf{withdraw}(t_1,t_2,t_3)) \;=\; \{\mathsf{cf}(\mathsf{withdraw}(t_1',t_2',\mathsf{valid}(t_1',t_2',t_3'))) \mid \bigwedge_{i=1..3} t_i' \in \tau_3^{\#}(t_i)\}$$

where the specification B4 not only forces all operations to be validated, but may also require an additional step to check the integrity of the account state before a debit operation is executed. This step is abstracted in an operation $\mathsf{cf} : Sys \longrightarrow Sys$. Therefore, operation withdraw is decomposed into a three step transaction:

spec B4
enrich INT
axioms
...
$\mathsf{bal}(\mathsf{deposit}(s, i, \mathsf{valid}(s, i, n)), i) \approx \mathsf{bal}(s, i) + \mathsf{valid}(s, i, n)$
$\mathsf{bal}(\mathsf{deposit}(s, i, n), i) \approx \mathsf{bal}(s, i) + n$
$\mathsf{bal}(\mathsf{cf}(\mathsf{withdraw}(s, i, \mathsf{valid}(s, i, n))), i) \approx \mathsf{max}(\mathsf{bal}(s, i) - \mathsf{valid}(s, i, n), 0)$ ◇

3.6. **Stepwise refinement revisited.** Having illustrated some typical applications of refinement by interpretation, it is legitimate to ask how does it relate to classical refinement based on signature morphisms. Let us first recall the standard definition:

Definition 3.14 (σ-Refinement)**.** Let $\sigma : \Sigma \to \Sigma'$ 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 $[\![SP']\!] \restriction_\sigma = \{A \restriction_\sigma | A \in [\![SP']\!]\}$. We write $SP \rightsquigarrow SP'$ whenever it is witnessed by the identity.

Note that, as we associate a fixed set of variables VAR to each signature, the usual notion of signature morphism has to be extended so that terms with variables from VAR can still be handled. Therefore we assume that each signature morphism $\sigma : \Sigma \to \Sigma'$ has a component σ_{var} on variables, namely an injective mapping from $\bigcup_{s \in S} \mathrm{VAR}_s$ to $\bigcup_{s \in S'} \mathrm{VAR}'_s$ such that, for every variable $x \in \mathrm{VAR}_s$, $\sigma_{var}(x) \in \mathrm{VAR}_{\sigma_{sort}(s)}$. This allows for the definition of σ^*, the extension of σ to terms given by $\sigma^*(x) = \sigma_{var}(x)$, for each variable x, and by $\sigma^*(f(t_1, \ldots, t_n)) = \sigma_{op}(f)(\sigma^*(t_1), \ldots, \sigma^*(t_n))$, for each term $f(t_1, \ldots, t_n)$ in Σ.

Since the composition of two signature morphisms is still a signature morphism, refinements can be composed vertically: if $SP_0 \rightsquigarrow_{\sigma_1} SP_1$ and $SP_1 \rightsquigarrow_{\sigma_2} SP_2$ then $SP_0 \rightsquigarrow_{\sigma_2 \circ \sigma_1} SP_2$. This is simply a consequence of reducts being functorial.

The so called *satisfaction lemma* [GB92] is at the basis of classical refinement. Actually, theorem 3.16 below, whose proof relies on the satisfaction lemma, provides an important characterisation of σ-refinements.

Lemma 3.15 (Satisfaction Lemma)**.** *Let Σ and Σ' be signatures, $\sigma : \Sigma \to \Sigma'$ a signature morphism, A' a Σ'-algebra, and ξ a conditional equation. Then,*

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

Theorem 3.16. *Let $\sigma : \Sigma \to \Sigma'$ be a signature morphism, $SP = \langle \Sigma, \Phi \rangle$ a X-flat specification and SP' a specification over Σ'. Then, $SP \rightsquigarrow_\sigma SP'$ iff $SP' \models \sigma(\Phi)$.*

Proof. Suppose that $SP \rightsquigarrow_\sigma SP'$. Then, for any $A' \in [\![SP']\!]$, $A' \restriction_\sigma \in [\![SP]\!]$, i.e., $A' \restriction_\sigma \models \Phi$. Hence, by Lemma 3.15, $A' \models \sigma(\Phi)$. On the other hand, suppose $SP' \models \sigma(\Phi)$. Then, for any $A' \in [\![SP']\!]$, $A' \models \sigma(\Phi)$. By Lemma 3.15 $A' \restriction_\sigma \models \Phi$, and hence, $A' \restriction_\sigma \in [\![SP]\!]$. Therefore $SP \rightsquigarrow_\sigma SP'$. □

A relationship between σ-refinement and refinement by interpretation can now be described as follows.

Theorem 3.17. *Let SP be a specification over Σ, and τ a translation from Σ to Σ' which interprets SP. If there is a specification SP^τ whose denotation coincides with $\mathrm{Mod}_\tau(SP)$, then, for every SP' over Σ', $SP^\tau \rightsquigarrow SP'$ implies $SP \multimap_\tau SP'$.*

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

Theorem 3.18. *Let SP be a specification over Σ and τ a translation from Σ to Σ'. If τ interprets SP and there is a specification SP^τ whose class of models coincides with $\mathrm{Mod}_\tau(SP)$, then the following conditions are equivalent:*

$$SP \multimap_\tau SP' \tag{3.1}$$

$$SP^\tau \rightsquigarrow SP' \tag{3.2}$$

Proof. Suppose that $SP \multimap_\tau SP'$. Since any $A \in [\![SP']\!]$ is a τ-model, $[\![SP']\!] \subseteq [\![SP^\tau]\!]$. Hence, $SP^\tau \rightsquigarrow SP'$. The converse implication is just Theorem 3.17. □

An immediate corollary is

Corollary 3.19. *Let $SP = \langle \Sigma, \Phi \rangle$ be a X-flat specification and τ a translation from Σ to Σ' which interprets SP and $\mathrm{Mod}_\tau(SP)$ is axiomatised by $\tau(\Phi)$. Then, $SP' \models \tau(\Phi)$ implies $SP \multimap_\tau SP'$.*

Proof. Let SP^τ be the flat specification $\langle \Sigma', \tau(\Phi) \rangle$. By hypothesis $[\![SP^\tau]\!] = \mathrm{Mod}_\tau(SP)$. Suppose that $SP' \models \tau(\Phi)$. Then $[\![SP']\!] \subseteq [\![SP^\tau]\!]$, which entails $SP^\tau \rightsquigarrow SP'$. By Theorem 3.17, since $[\![SP^\tau]\!] = \mathrm{Mod}_\tau(SP)$, we conclude $SP \multimap_\tau SP'$. □

This paves the way to address the following question: *given that any mapping can be regarded as a multifunction, when does a σ-refinement via signature morphism become a refinement by interpretation?*

First note that a signature morphism $\sigma : \Sigma \to \Sigma'$ induces a translation $\tau : \mathrm{Ceq}(\Sigma) \to \mathrm{Ceq}(\Sigma')$. This is defined, for each $\langle \Gamma, e \rangle \in \mathrm{Ceq}(\Sigma)$, by $\tau(\langle \Gamma, t \approx t' \rangle) = \langle \{\sigma(t) \approx \sigma(t') | t \approx t' \in \Gamma\}, \sigma(t) \approx \sigma(t') \rangle$. Then,

Lemma 3.20. *Let SP be a specification over Σ, $\sigma : \Sigma \to \Sigma'$ an injective signature morphism, and τ the translation induced by the signature morphism σ. Then τ interprets SP.*

Proof. Let $K = \{A' | A' \restriction_\sigma \in [\![SP]\!]\}$. Let $\xi \in \mathrm{Ceq}(\Sigma)$, and suppose $SP \models \xi$. Let $A' \in K$. Since $A' \restriction_\sigma \in [\![SP]\!]$ we have that $A' \restriction_\sigma \models \xi$ and, by Lemma 3.15, $A' \models \sigma(\xi)$. Therefore $K \models \tau(\xi)$. Suppose now $K \models \tau(\xi)$ and let $A \in [\![SP]\!]$. Since σ is injective there is $B \in K$ such $B \restriction_\sigma = A$. Thus $B \models \tau(\xi)$ and, by Lemma 3.15, $B \restriction_\sigma \models \xi$, *i.e.*, $A \models \xi$. Hence $SP \models \xi$. Therefore σ interprets SP. □

Next theorem shows that refinement by interpretation is, actually, a generalisation of σ-refinement whenever the signature morphism σ is injective.

Theorem 3.21. *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 $SP \multimap_\tau SP'$.*

Proof. By the previous theorem τ interprets SP. Suppose $SP \rightsquigarrow_\sigma SP'$, *i.e.*, $[\![SP']\!]\restriction_\sigma \subseteq [\![SP]\!]$. Let $\xi \in \mathrm{Ceq}(\Sigma)$ such that $SP \models \xi$. Let $A' \in [\![SP']\!]$. Then $A'\restriction_\sigma \in [\![SP]\!]$ and so $A'\restriction_\sigma \models \xi$. By Lemma 3.15, $A' \models \sigma(\xi)$. Hence $SP' \models \sigma(\xi)$. Therefore $SP \rightarrow_\tau SP'$. □

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

Theorem 3.22. *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 $SP \rightarrow_\tau SP'$.*

Proof. Suppose $SP \rightarrow_\tau SP'$. Since $SP \models \Phi$, $SP' \models \sigma(\Phi)$. By Theorem 3.16 $SP \rightsquigarrow_\sigma SP'$. □

It should be noted at this point that the discussion concerning composition of refinements by interpretation is not straightforward. In fact, *horizontal composition* is still an open question (see section 6). For *vertical composition* an additional property has to be imposed on the components' interpretations. Formally,

Theorem 3.23. *Let SP, SP' and SP'' be three specifications over Σ, Σ' and Σ'' respectively. Let τ be a translation from Σ to Σ' and ρ a translation from Σ' to Σ'' Suppose that $SP \rightarrow_\tau SP'$, $SP' \rightarrow_\rho SP''$ and that there exists a specification SP^τ over Σ' such that $[\![SP^\tau]\!] = \mathrm{Mod}_\tau(SP)$ and ρ interprets SP^τ. Then $SP \rightarrow_{\rho\circ\tau} SP''$.*

Proof. Since τ interprets SP, $\mathrm{Mod}_\tau(SP)$ also interprets SP. Let $\xi \in \mathrm{Ceq}(\Sigma)$. Then $SP \models \xi \Leftrightarrow \mathrm{Mod}_\tau(SP) \models \tau(\xi) \Leftrightarrow SP^\tau \models \tau(\xi)$. But, since ρ interprets SP^τ, this is equivalent to $K \models \rho(\tau(\xi))$, for some class K of Σ''-algebras. Thus, $\rho \circ \tau$ interprets SP. On the other hand, suppose $SP \models \xi$. Since $SP \rightarrow_\tau SP'$, $SP' \models \tau(\xi)$. And, from $SP' \rightarrow_\rho SP''$ we have $SP'' \models \rho(\tau(\xi))$. Therefore, putting everything together, $SP \rightarrow_{\rho\circ\tau} SP''$.

□

3.7. **Going generic.** This section introduced refinement by interpretation in the specific setting of algebraic specification over the institution of Horn clause logic. Our discussion built on the fact that each specification which is flat, or equivalent to a flat specification, induces a 2-dimension deductive system corresponding to its class of models. This made possible the use of interpretations to reason about the refinement of specifications.

A similar discussion could have been made directly over 2-dimension deductive systems. Revisiting specification T in example 3.1 in such setting will lead to the deductive system depicted in Figure 7. On its turn, specification S in the same example corresponds to the deductive system EQ_Σ for the relevant signature Σ (see Figure 1 in Subsection 2.3). Note that we are now considering an arbitrary binary predicate and not necessarily the equality relation. It may stand, for example, for bisimilarity or other form of observational equivalence. This explains the need for the explicit introduction of axioms and inference rules which would otherwise be assumed for equality.
Clearly the translation

$$\langle x, x' \rangle \;\mapsto\; \langle test(x, x'), ok \rangle$$

interprets EQ_Σ. In the sequel, this generalisation will be carried on further, leading to a theory of refinement by interpretation over arbitrary k-dimensional deductive systems. We will resort to a CASL-inspired notation to describe (finitary) deductive systems (coming from algebraic specifications or not), as illustrated in Figure 7.

deductive system DT
sorts
s
ops
$ok :\longrightarrow s$
$f : s \longrightarrow s$
$test : s \times s \longrightarrow s$

axioms
$\langle test(x, x), ok\rangle$

inference rules

$$\frac{\langle test(x, x'), ok\rangle \quad \langle test(x', x''), ok\rangle}{\langle test(x, x''), ok\rangle}$$

$$\frac{\langle test(x, x'), ok\rangle}{\langle test(x', x), ok\rangle} \qquad \frac{\langle test(x, x'), ok\rangle}{\langle test(f(x), f(x')), ok\rangle}$$

Figure 7: Deductive system DT

4. Logical interpretation in a general setting

This section generalises translations and logical interpretations to arbitrary k-dimensional deductive systems. In particular, the case of k-dimensional systems possessing an algebraic semantics is discussed in some detail.

4.1. **Translations.** The first step to generalise *refinement by interpretation* from the equational case is to define the notion of *translation* in the general setting of k-dimensional deductive systems. The following definition generalises Definition 3.2, still assuming that, for each signature, the set VAR of variables is locally countably infinite.

Definition 4.1 (Translation)**.** Let Σ and Σ' be two signatures. A (k, l)-*translation from* Σ *to* Σ' is a globally finite sorted *multifunction* $\tau : \mathrm{Fm}^k(\Sigma) \preccurlyeq \mathrm{Fm}^l(\Sigma')$, i.e., for any $s \in S$ and $\bar{\varphi} \in \mathrm{Fm}^k(\Sigma)_s$, $\tau_s(\bar{\varphi})$ is a globally finite S'-sorted set of l-formulas over Σ'.

As before, τ is called a *self translation* of Σ whenever Σ and Σ' coincide. In this case, we say that τ *commutes with substitutions* if for every substitution σ and every formula $\bar{\varphi} \in \mathrm{Fm}^k(\Sigma)$ $\tau(\sigma(\bar{\varphi})) = \sigma(\tau(\bar{\varphi}))$. The translation can be specified by giving, for each sort s, the image $\tau_s(\bar{x}{:}s)$ for a k-variable $\bar{x}{:}s$ (see Example 4.4). Given a (k, l)-translation τ and an inference rule $\xi = \langle \Gamma, \bar{\varphi}\rangle$, we write $\tau(\xi)$ for the set of inference rules $\{\langle \tau(\Gamma), \bar{\psi}\rangle : \bar{\psi} \in \tau(\bar{\varphi})\}$.

A self (k, l)-translation τ is *schematic* if there is a S-sorted set Δ of l-formulas, where for each s, $\Delta_s(\bar{x})$ is a set of l-formulas over Σ' in the k-variable $\langle x_0 : s, \ldots, x_{n-1} : s\rangle$ such that, for any $\bar{\varphi} \in \mathrm{Fm}^k(\Sigma)_s$, $\tau_s(\bar{\varphi}) = \Delta_s(\varphi_0, \ldots, \varphi_{k-1})$.We say that a (k, l)-translation is *functional* if the image of each k-formula is a singleton. Schematic (2,2)-translations were

deductive system SLV
enrich EQ_Σ
axioms
$\langle p, p \wedge p \rangle$
$\langle p \wedge q, q \wedge p \rangle$
$\langle p \wedge (q \wedge r), (p \wedge q) \wedge r \rangle$

Figure 8: Semilattices as algebras.

first used in [Mad08]. Finally, the following result is the obvious generalisation of Lemma 3.3 from Section 3.

Lemma 4.2. *Let Σ be a standard signature and τ a self (k,l)-translation of Σ. Then the following conditions are equivalent:*

(i) *τ commutes with substitutions.*
(ii) *There exists a k-variable $\bar{x} = \langle x_0, \ldots, x_{k-1} \rangle$ and a S-sorted set $\Delta(\bar{x})$ of l-formulas in $\bar{x}$ such that, for any $\bar{\varphi} \in \mathrm{Fm}^k(\Sigma)_s$, $\tau_s(\bar{\varphi}) = \Delta_s(\bar{\varphi})$.*

4.2. **Interpretations.** Similarly to the equational case not all translations lead to refinements. Hence, we start by generalising the definition of *interpretation*.

Definition 4.3 (Interpretation)**.** Let τ be a (k,l)-translation from Σ to Σ', and $\mathcal{L}$ a k-deductive system over Σ. We say that τ *interprets* $\mathcal{L}$ if there is a l-deductive system $\mathcal{L}'$ over Σ' such that, for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$, $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ if and only if $\tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi})$. In this case we say that τ *interprets* $\mathcal{L}$ *in* $\mathcal{L}'$ and $\mathcal{L}'$ *is a τ-interpretation of* $\mathcal{L}$.

To illustrate this more general notion of an interpretation, consider the following examples which capture a change of logic paradigm. Integrating such a move in the refinement process, by witnessing refinement steps with this sort of interpretations, was, from the outset, the motivation for this generalisation.

Example 4.4 (CPC vs. Boolean algebras)**.** The deductive system encoding the equational logic of Boolean algebras $\mathcal{L}_{\mathrm{BA}}$ interprets classical propositional logic (CPC), both over the one-sorted signature $\Sigma = \{\rightarrow, \wedge, \vee, \neg, \top, \bot\}$, under the schematic, self (1,2)-translation $\tau(p) = \{\langle p, \top \rangle\}$. Moreover, the deductive system $\mathcal{L}_{\mathrm{HA}}$, induced by the class of Heyting algebras HA, also provides an interpretation of CPC under the translation $\nu(p) = \{\langle \neg\neg p, \top \rangle\}$. This translation also interprets CPC into $\mathcal{L}_{\mathrm{BA}}$ which shows that an interpretation may not be unique [BR03].

Reciprocally, as one would expect, CPC also interprets $\mathcal{L}_{\mathrm{BA}}$, under the (2,1)-translation $\rho(\langle p, q \rangle) = \{p \rightarrow q, q \rightarrow p\}$. ◇

Example 4.5 (Semilattices into posets)**.** A semilattice can be regarded either as an algebra or as a partial order structure. Such a duality, often useful in specifications, can be expressed, in a natural way, by an interpretation, actually an equivalence between two 2-deductive systems over the one-sorted signature $\Sigma = \{\wedge\}$ (see [BP01]) depicted in Figures 8 and 9. The schematic translation τ, defined by the multifunction $\tau(\langle p, q \rangle) = \{\langle p, q \rangle, \langle q, p \rangle\}$, witnesses the interpretation of SLV by SLP.

deductive system SLP
axioms

$\langle p, p\rangle$
$\langle p, p \wedge p\rangle$
$\langle p \wedge q, p\rangle$
$\langle p \wedge q, q\rangle$

inference rules

$$\frac{\langle x, y\rangle, \langle y, z\rangle}{\langle x, z\rangle}$$

$$\frac{\langle x_0, y_0\rangle, \langle x_1, y_1\rangle}{(x_0 \wedge x_1, y_0 \wedge y_1\rangle}$$

Figure 9: Semilattices as order structures.

◇

Clearly, interpretations compose in the sense that if $\mathcal{L}'$ is a τ-interpretation of $\mathcal{L}$ and $\mathcal{L}''$ is a ρ-interpretation of $\mathcal{L}'$ then $\rho \circ \tau$ interprets $\mathcal{L}$ in $\mathcal{L}''$. Other properties need further investigation. The following subsection explores a class of interpretations specifically relevant for software design.

4.3. **Towards an algebraic semantics.** There are k-deductive systems to which an algebraic specification can be associated, thus providing an alternative semantics (called *algebraic semantics* in the context of algebraic logic). It is well known [BR03] that this association is not unique and may not exist. Our investigation starts with the following definition which generalises Definition 3.7:

Definition 4.6 (τ-model)**.** Let τ be a (k, l)-translation from Σ to Σ' and $\mathcal{L}$ a k-deductive system over Σ. An l-structure $\mathcal{A}$ is a *τ-model of $\mathcal{L}$* if for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$, $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ implies $\tau(\Gamma) \models_{\mathcal{A}} \tau(\bar{\varphi})$. The class of all τ-models of $\mathcal{L}$, denoted by $\mathrm{Mod}_\tau(\mathcal{L})$, is called the *$\tau$-model class of $\mathcal{L}$*.

As mentioned above, the semantic consequence associated to a class of k-structures defined over $\mathrm{Fm}^k(\Sigma)$, is always a k-deductive system even if it fails to be specifiable. Hence, $\models_{\mathrm{Mod}_\tau(\mathcal{L})}$ is a deductive system which we will denote by $\mathcal{L}^\tau$. Furthermore,

Theorem 4.7. *Let τ be a (k, l)-translation from Σ to Σ' and $\mathcal{L}$ a k-deductive system over Σ. If τ interprets $\mathcal{L}$, then the l-deductive system $\mathcal{L}^\tau$ is a τ-interpretation of $\mathcal{L}$; moreover, this is the τ-interpretation of $\mathcal{L}$ with the largest class of models.*

Proof. Suppose that τ interprets $\mathcal{L}$. Let $\mathcal{L}'$ be a specification which is a τ-interpretation of $\mathcal{L}$. Then for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$, $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ iff $\tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi})$ iff $\tau(\Gamma) \models_{\mathrm{Mod}(\mathcal{L}')} \tau(\bar{\varphi})$. Hence all models of $\mathcal{L}'$ are τ-models of $\mathcal{L}$. Thus, $\mathrm{Mod}(\mathcal{L}') \subseteq \mathrm{Mod}_\tau(\mathcal{L})$.
So, it is enough to prove that $\mathcal{L}^\tau$ is a τ-interpretation of $\mathcal{L}$. Let $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$. It is clear that $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ implies $\tau(\Gamma) \vdash_{\mathcal{L}^\tau} \tau(\bar{\varphi})$. Suppose now that $\tau(\Gamma) \vdash_{\mathcal{L}^\tau} \tau(\bar{\varphi})$. Let $\mathcal{L}'$

be a specification that is a τ-interpretation of $\mathcal{L}$ (it exists since τ interprets $\mathcal{L}$). Since, $\mathrm{Mod}(\mathcal{L}') \subseteq \mathrm{Mod}_\tau(\mathcal{L})$, $\tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi})$. Thus $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ because $\mathcal{L}'$ is a τ-interpretation of $\mathcal{L}$. □

Our focus on algebraic specifications entails the need for paying special attention to $(k,2)$-translations, which, mapping k-formulas to 2-formulas, provide a way to relate an arbitrary k-deductive system to a suitable class of algebras. For the remaining of this section we will consider a 2-formula $\langle t, t' \rangle$ as an equation $t \approx t'$.

Let τ be a $(k,2)$-translation from Σ to Σ' and $\mathcal{L}$ a k-deductive system. A class K of Σ-algebras is said to be a *τ-algebraic semantics of $\mathcal{L}$* if τ interprets $\mathcal{L}$ in $\models_K$. Thus, we define the algebraic model class $K^\tau_{\mathcal{L}}$ over Σ' as the class of algebraic reducts of the τ-models $\mathcal{L}$ taking the identity as a filter. Formally,

$$K^\tau_{\mathcal{L}} \;=\; \{A | \langle A, \triangle_A \rangle \text{ is a } \tau\text{-model}\},$$

which paves the way to the following corollary:

Corollary 4.8. *Given a $(k,2)$-translation τ from Σ to Σ', and a k-deductive system $\mathcal{L}$ over Σ, if there is a τ-algebraic semantics of $\mathcal{L}$, then the class $K^\tau_{\mathcal{L}}$ is the largest τ-algebraic semantics of $\mathcal{L}$,* i.e.*, with the largest class of models. Moreover, $K^\tau_{\mathcal{L}}$ is finitely axiomatised whenever $\mathcal{L}$ is finitely axiomatisable.*

In practice, however, it may happen that $K^\tau_{\mathcal{L}}$ is too wide for the envisaged purposes, namely to discuss implementations. The following theorem gives a sufficient and necessary condition for a subclass of $K^\tau_{\mathcal{L}}$ to be a τ-algebraic semantics of $\mathcal{L}$. Similar results are well known for sentential logics [BR03]. In this paper, however, we reformulate them for k-dimensional and many sorted logics, since they are a vehicle to sufficient and necessary conditions for a deductive system to have an algebraic semantics. Consider, therefore, the mapping $\tau_{\mathcal{L},K} : \mathrm{Th}(\mathcal{L}) \to \mathrm{Th}(K)$ defined by $\tau_{\mathcal{L},K}(T) = \mathrm{Cn}_K(\tau(T))$, for all $T \in \mathrm{Th}(\mathcal{L})$. Then,

Lemma 4.9. *Let $\mathcal{L}$ be a deductive system, τ a self $(k,2)$-translation of Σ which commutes with substitutions and $K \subseteq K^\tau_{\mathcal{L}}$. The following conditions are equivalent:*

(i) *K is a τ-algebraic semantics of $\mathcal{L}$.*
(ii) *$\tau_{\mathcal{L},K}$ is injective.*

Proof. Let $T_1, T_2 \in \mathrm{Th}(\mathcal{L})$ and $\bar{\alpha} \in T_1$. Suppose $\tau_{\mathcal{L},K}(T_1) = \tau_{\mathcal{L},K}(T_2)$. We have that $\tau(\bar{\alpha}) \subseteq \tau(T_1) \subseteq \tau_{\mathcal{L},K}(T_1) = \tau_{\mathcal{L},K}(T_2)$, *i.e.*, $\tau(T_2) \models_K \tau(\bar{\alpha})$. Since K is an τ-algebraic semantics of $\mathcal{L}$, $T_2 \vdash_{\mathcal{L}} \bar{\alpha}$, *i.e.*, $\bar{\alpha} \in T_2$. Thus $T_1 \subseteq T_2$. Similarly, we can prove that $T_2 \subseteq T_1$. We conclude that $\tau_{\mathcal{L},K}$ is injective.

Conversely, let $\Gamma \cup \{\bar{\alpha}\} \subseteq \mathrm{Fm}^k(\Sigma)$. Since K is a class of algebraic reducts of τ-models of $\mathcal{L}$, we have that $\Gamma \vdash_{\mathcal{L}} \bar{\alpha}$ implies $\tau(\Gamma) \models_K \tau(\bar{\alpha})$. Now, suppose $\tau(\Gamma) \models_K \tau(\bar{\alpha})$. Thus, $\mathrm{Cn}_K(\tau(\Gamma)) = \mathrm{Cn}_K(\tau(\Gamma \cup \{\bar{\alpha}\}))$. Since $\Gamma \subseteq \mathrm{Cn}_{\mathcal{L}}(\Gamma)$, we have that $\tau(\Gamma) \subseteq \tau(\mathrm{Cn}_{\mathcal{L}}(\Gamma))$. Thus $\mathrm{Cn}_K(\tau(\Gamma)) \subseteq \mathrm{Cn}_K(\tau(\mathrm{Cn}_{\mathcal{L}}(\Gamma))) = \tau_{\mathcal{L},K}(\mathrm{Cn}_{\mathcal{L}}(\Gamma))$. To prove the reverse inclusion, let $t \approx t' \in \tau_{\mathcal{L},K}(\mathrm{Cn}_{\mathcal{L}}(\Gamma))$.Thus $\{\tau(\xi) : \Gamma \vdash_{\mathcal{L}} \xi\} \models_K t \approx t'$. Again, since K is a class of algebraic reducts of τ-models of $\mathcal{L}$, for all $t \approx t' \in \mathrm{Fm}^2(\Sigma)$, we have that $\Gamma \vdash_{\mathcal{L}} \xi$ implies $\tau(\Gamma) \models_K \tau(\xi)$. Hence $\tau(\Gamma) \models_K t \approx t'$, i.e., $t \approx t' \in \mathrm{Cn}_K(\tau(\Gamma))$. Therefore, for all $\Gamma \subseteq \mathrm{Fm}^k(\Sigma)$, $\tau_{\mathcal{L},K}(\mathrm{Cn}_{\mathcal{L}}(\Gamma)) = \mathrm{Cn}_K(\tau(\Gamma))$. Thus, $\tau_{\mathcal{L},K}(\mathrm{Cn}_{\mathcal{L}}(\Gamma)) = \tau_{\mathcal{L},K}(\mathrm{Cn}_{\mathcal{L}}(\Gamma \cup \{\bar{\alpha}\}))$. Since $\tau_{\mathcal{L},K}$ is injective, $\mathrm{Cn}_{\mathcal{L}}(\Gamma) = \mathrm{Cn}_{\mathcal{L}}(\Gamma \cup \{\bar{\alpha}\})$, i.e., $\Gamma \vdash_{\mathcal{L}} \bar{\alpha}$. □

Lemma 4.9 and the fact that class $K^{\tau}_{\mathcal{L}}$ is a τ-algebraic semantics, entail another important result: if $\mathcal{L}$ has a τ-algebraic semantics, then any extension of $\mathcal{L}$ also has a τ-algebraic semantics, for τ a self $(k,2)$-translation of Σ commuting with substitutions. This is recorded in Theorem 4.11 below, whose proof requires the following lemma.

Lemma 4.10. *Let $\mathcal{L}$ be a specifiable k-deductive system and τ a self $(k,2)$-translation of Σ which commutes with substitutions. Suppose $K = K^{\tau}_{\mathcal{L}}$ is an τ-algebraic semantics of $\mathcal{L}$. If $\mathcal{L}'$ is an extension of $\mathcal{L}$, and $K' = K^{\tau}_{\mathcal{L}}$ then $\tau_{\mathcal{L}',K'}$ equals $\tau_{\mathcal{L},K}$ restricted to* $\mathrm{Th}(\mathcal{L}')$.

Proof. Let $T \in \mathrm{Th}(\mathcal{L}')$. Since $\mathcal{L}'$ is an extension of $\mathcal{L}$, $K' \subseteq K$, and $\models_{K'}$ is an extension of $\models_K$. Hence $\mathrm{Cn}_K(\tau[T]) \subseteq \mathrm{Cn}_{K'}(\tau[T])$, *i.e.*, $\tau_{\mathcal{L},K}[T] \subseteq \tau_{\mathcal{L}',K'}[T]$. For the reverse inclusion note that K' can be axiomatised by a set of axioms and a set of inference rules. It is not difficult to see that $\mathrm{Cn}_K(\tau[T])$ contains all substitution instances of the axioms of $\models_{K'}$ and is closed under the inference rules of $\models_{K'}$. Let $\alpha \in \mathrm{Thm}(\mathcal{L}')$ and e be a substitution. Since $\mathrm{Thm}(\mathcal{L}')$ is closed under substitutions, $\vdash_{\mathcal{L}'} e(\alpha)$. Thus for all $T \in \mathrm{Th}(\mathcal{L}')$, we have that $e(\alpha) \in T$. As τ commutes with arbitrary substitutions, $e[\tau(\alpha)] = \tau[e(\alpha)] \subseteq \tau[T] \subseteq \mathrm{Cn}_K(\tau[T])$. Thus $\mathrm{Cn}_K(\tau[T])$ contains all substitution instances of axioms of $\models_{K'}$. Now, let $\{\alpha_i : i < n\} \vdash_{\mathcal{L}'} \beta$ be an inference rule of $\mathcal{L}'$ and e a substitution such that $\{e[\tau(\alpha_i)] : i < n\} \subseteq \mathrm{Cn}_K(\tau[T])$, i.e., $\tau[T] \models_K e[\tau(\alpha_i)]$ for all $i < n$. Since τ commutes with arbitrary substitutions, $e[\tau(\alpha_i)] = \tau[e(\alpha_i)]$ for all $i < n$, *i.e.*, $\tau[T] \models_K \tau[e(\alpha_i)]$, for all $i < n$. As K is an algebraic semantics of $\mathcal{L}$, it follows that $T \vdash_S e(\alpha_i)$ for all $i < n$, *i.e.*, $\{e(\alpha_i) : i < n\} \subseteq T$. By structurality of $\mathcal{L}'$, $\{e(\alpha_i) : i < n\} \vdash_{\mathcal{L}'} e(\beta)$. Since $T \in \mathrm{Th}(\mathcal{L}')$, we have that $e(\beta) \in T$. Thus $e[\tau(\beta)] = \tau[e(\beta)] \subseteq \tau[T] \subseteq \mathrm{Cn}_K(\tau[T])$. Therefore $\mathrm{Cn}_K(\tau[T])$ is closed under the inference rules of K'. By the characterisation of a theory in a deductive system, we have proved that $\mathrm{Cn}_K(\tau[T]) \in \mathrm{Th}(K')$. Since $\tau[T] \subseteq \mathrm{Cn}_K(\tau[T])$ and $\mathrm{Cn}_{K'}(\tau[T])$ is the least $\mathcal{L}'$-theory that contains $\tau[T]$, we have that $\mathrm{Cn}_{K'}(\tau[T]) \subseteq \mathrm{Cn}_K(\tau[T])$, i.e., $\tau_{\mathcal{L}',K'}[T] \subseteq \tau_{\mathcal{L},K}[T]$. □

The following main result can now be proved:

Theorem 4.11. *Let $\mathcal{L}$ be a specifiable k-deductive system and τ a self $(k,2)$-translation of Σ which commutes with substitutions. If $\mathcal{L}$ has an τ-algebraic semantics, then any extension of $\mathcal{L}$ has a τ-algebraic semantics as well.*

Proof. Let K be a τ-algebraic semantics of $\mathcal{L}$. Let $\mathcal{L}'$ be an extension of $\mathcal{L}$ and $K' = K^{\tau}_{\mathcal{L}'}$. By Corollary 4.8, the class $K^{\tau}_{\mathcal{L}}$ is a τ-algebraic semantics. Since, by Lemma 4.9, the mapping $\tau_{\mathcal{L},K}$ is injective, we have, by Lemma 4.10, that the mapping $\tau_{\mathcal{L}',K'}$ is also injective. Again by Lemma 4.9, K' is a τ-algebraic semantics of $\mathcal{L}'$. □

5. Refinement by interpretation: The general case

This section revisits the notion of refinement by interpretation in the general setting of arbitrary k-dimensional deductive systems, and illustrates the design flexibility it entails.

As before, our concern is to put forward a precise, but flexible notion of what counts for a valid refinement step in software design. Our starting point is the following syntactic-grounded notion,

Definition 5.1 ((Logical) refinement)**.** Let Σ and Σ' be two signatures such that $\Sigma \subseteq \Sigma'$, and $\mathcal{L}, \mathcal{L}'$ two k-deductive systems over Σ and Σ', respectively. We say that $\mathcal{L}'$ is a *(logical) refinement of* $\mathcal{L}$, in symbols $\mathcal{L} \rightsquigarrow \mathcal{L}'$, if for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$,

$$\Gamma \vdash_{\mathcal{L}} \bar{\varphi} \Rightarrow \Gamma \vdash_{\mathcal{L}'} \bar{\varphi}.$$

Note that, when $\mathcal{L}$ is specifiable, $\mathcal{L} \rightsquigarrow \mathcal{L}'$ if all the axioms of $\mathcal{L}$ are theorems of $\mathcal{L}'$ and the theories of $\mathcal{L}'$ are compatible with the inference rules of $\mathcal{L}$.

Example 5.2. Modal logic $\mathrm{S5^G}$ forms a (logical) refinement of CPC. Consider the modal signature $\Sigma = \{\rightarrow, \wedge, \vee, \neg, \top, \bot, \Box\}$. Modal logic K is obtained from CPC by adding the symbol $\Box$ to the signature, the axiom $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ and the inference rule $\dfrac{p}{\Box p}$. Logic $\mathrm{S5^G}$, on the other hand, enriches the signature of K with the symbol $\Diamond$, and K itself with the axioms $\Box p \rightarrow p$, $\Box p \rightarrow \Box\Box p$ and $\Diamond p \rightarrow \Box\Diamond p$ [BP01]. Hence, since the signature of both systems contains the signature of CPC and their presentations result from the introduction of extra axioms and inference rules to the CPC presentation, we have, by the previous fact that CPC $\rightsquigarrow$ K and CPC $\rightsquigarrow \mathrm{S5^G}$ (actually, CPC $\rightsquigarrow$ K $\rightsquigarrow \mathrm{S5^G}$). Thus, refining CPC in this way, we acquire enough expressivity to state properties over propositions like *it is necessary that* ϕ (by $\Box\phi$) and *it is possible that* ϕ (by $\Diamond\phi$). This kind of refinement makes possible the accommodation of a new type of requirements, modally expressed, along the refinement process. $\Diamond$

Theorem 5.3. *Let Σ be a signature and $\mathcal{L}$ and $\mathcal{L}'$ two k-deductive systems over Σ. Then the following conditions are equivalent*

(i) $\mathcal{L} \rightsquigarrow \mathcal{L}'$
(ii) $\mathrm{Mod}(\mathcal{L}') \subseteq \mathrm{Mod}(\mathcal{L})$.

Proof. (i) $\Rightarrow$ (ii). Suppose $\mathcal{L} \rightsquigarrow \mathcal{L}'$. Let $\mathcal{A} \in \mathrm{Mod}(\mathcal{L}')$ and $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$. Suppose $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$. We have by (i) that $\Gamma \vdash_{\mathcal{L}'} \bar{\varphi}$ and hence $\Gamma \vdash_{\mathcal{A}} \bar{\varphi}$. Therefore $\mathcal{A} \in \mathrm{Mod}(\mathcal{L})$.
ii) $\Rightarrow$ i). Suppose $\mathrm{Mod}(\mathcal{L}') \subseteq \mathrm{Mod}(\mathcal{L})$. Let $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$. Suppose $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$. Let $\mathcal{A} \in \mathrm{Mod}(\mathcal{L}')$. By ii) we have $\mathcal{A} \in \mathrm{Mod}(\mathcal{L})$ and hence $\Gamma \vdash_{\mathcal{A}} \bar{\varphi}$. Therefore, by Completeness, $\Gamma \vdash_{\mathcal{L}'} \bar{\varphi}$. $\square$

A coarser and more flexible definition of refinement, however, is provided by the notion of logical interpretation, as already shown in the equational case. Formally,

Definition 5.4 (Refinement by interpretation)**.** Let $\mathcal{L}$ be a k-deductive system over Σ and τ a (k,l)-translation from Σ to Σ', which interprets $\mathcal{L}$. We say that a l-deductive system $\mathcal{L}'$ over Σ' *refines the deductive system* $\mathcal{L}$ *via the interpretation* τ, in symbols $\mathcal{L} \rightarrowtail_{\tau} \mathcal{L}'$, if for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$,

$$\Gamma \vdash_{\mathcal{L}} \bar{\varphi} \;\Rightarrow\; \tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi}).$$

The requirement that τ has to interpret $\mathcal{L}$ is necessary in order to enforce some control over the class of models of the deductive system $\mathcal{L}'$. In particular, this guarantees that $\mathrm{Mod}(\mathcal{L}')$ has to be smaller than $\mathrm{Mod}_{\tau}(\mathcal{L})$.

The following two examples illustrate this general notion of refinement at work.

Example 5.5. Any subclass of the class of Boolean algebras induces a refinement by interpretation of CPC, based on the usual (1,2)-translation τ given by $\tau(p) = \{\langle p, \top\rangle\}$). $\Diamond$

deductive system ORDBAMS
axioms

$\langle n, n\rangle \;\; \langle i, i\rangle \;\; \langle x, x\rangle$
$\langle n, n + 0\rangle$
$\langle n + 0, n\rangle$
$\langle n + n', n' + n\rangle$
...
$\langle \mathsf{bal}(\mathsf{deposit}(x, i, n), i) \;,\; \mathsf{bal}(x, i) + n\rangle$
$\langle \mathsf{bal}(x, i) + n \;,\; \mathsf{bal}(\mathsf{deposit}(x, i, n), i)\rangle$
$\langle \mathsf{bal}(\mathsf{withdraw}(x, i, n), i) \;,\; \mathsf{max}(\mathsf{bal}(x, i) - n, 0)\rangle$
$\langle \mathsf{max}(\mathsf{bal}(x, i) - n, 0) \;,\; \mathsf{bal}(\mathsf{withdraw}(x, i, n), i)\rangle$
...

inference rules

$$\frac{\langle n, n'\rangle \; \langle n', n\rangle}{\langle n, n''\rangle}$$

$$\frac{\langle n, n'\rangle}{\langle n + n'', n' + n''\rangle} \qquad \frac{\langle n, n'\rangle}{\langle -n', -n\rangle} \qquad \frac{\langle n, n'\rangle}{\langle s(n), s(n')\rangle}$$

$$\frac{\langle x, y\rangle \; \langle i, j\rangle}{\langle \mathsf{bal}(x, i), \mathsf{bal}(y, j)\rangle} \qquad \frac{\langle x, y\rangle \; \langle i, j\rangle}{\langle \mathsf{bal}(y, j), \mathsf{bal}(x, i)\rangle}$$

$$\frac{\langle x, y\rangle \; \langle i, j\rangle \; \langle n, n'\rangle \; \langle n', n\rangle}{\langle \mathsf{deposit}(x, i, n), \mathsf{deposit}(y, j, n')\rangle}$$

$$\frac{\langle x, y\rangle \; \langle i, j\rangle \; \langle n, n'\rangle \; \langle n', n\rangle}{\langle \mathsf{withdraw}(x, i, n), \mathsf{withdraw}(y, i, n)\rangle}$$

...

$$\frac{\langle x, y\rangle}{\langle y, x\rangle}$$

Figure 10: Revisiting BAMS.

Example 5.6. Consider the fragment of the specification BAMS, of a toy bank account management system, given in Example 3.13, regarded as a 2-deductive system. Suppose we intend to refine this system by imposing that the *balance of each account has to be positive*. This cannot be easily expressed in (strict) equational logic. However, it can be captured as a refinement by interpretation. Actually, consider the 2-deductive system over Σ sketched in Figure 10, in which n, n', n'' are variables of sort Int, x, y of sort Sys and i, j of sort Ac. Notice that only a few axioms and inference rules are shown for illustration purposes. Intuitively we intend to interpret differently the binary predicates: as equality for the carriers of Ac and Sys; as $\leq$ for the integers.

Let us take a fixed-semantics approach by fixing the Int component as the integers endowed with the usual operations. Consider, thus, the following subclasses of the model class of BAMS and ORDBAMS, respectively:

$$\begin{aligned}\mathrm{BAMS}^{\mathbb{Z}} &= \{\langle A, F\rangle \in \mathrm{Mod}(\mathrm{BAMS}) : A_{Int} = \mathbb{Z} \,\&\, F_{Int} = id_{\mathbb{Z}}\}\\ \mathrm{ORDBAMS}^{\mathbb{G}} &= \{\langle A, G\rangle \in \mathrm{Mod}(\mathrm{ORDBAMS}) : A_{Int} = \mathbb{Z} \,\&\, G_{Int} = \leq\}\end{aligned}$$

Notice that a structure $\langle A, F\rangle$ reduces to an algebra when, for each sort, the corresponding filter in F is the identity. Let τ be the (2,2)-translation defined schematically by

$$\begin{aligned}\tau_{Int}(\langle n, n'\rangle) &= \{\langle n, n'\rangle, \langle n', n\rangle\}\\ \tau_{Ac}(\langle i, j\rangle) &= \{\langle i, j\rangle\}\\ \tau_{Sys}(\langle x, y\rangle) &= \{\langle x, y\rangle\}\end{aligned}$$

The underlying intuition is that an equation $n \approx n'$ of sort Int is translated into two inequalities $n \leq n'$ and $n \leq n'$. Clearly, $\models_{\mathrm{ORDBAMS}^{\mathbb{Z}}}$ is a τ-interpretation of $\models_{\mathrm{BAMS}^{\mathbb{Z}}}$. Therefore, the deductive system which extends ORDBAMS to capture the extra requirement $\langle 0, \mathsf{bal}(x, i)\rangle$ is obtained as a τ-refinement of the original one. ◇

We complete the discussion of refinement by interpretation in this general setting by establishing its connection to classical, signature morphism based refinement. What follows lifts the corresponding discussion in subsection 3.6 to the level of k-deductive systems and their interpretations:

Theorem 5.7. *Let $\mathcal{L}$ and $\mathcal{L}'$ be a k-deductive system over Σ and l-deductive system over Σ' respectively. Let τ be a (k, l)-translation from Σ to Σ'. Then the following conditions are equivalent*

(i) $\mathcal{L} \rightharpoonup_\tau \mathcal{L}'$;
(ii) *$\mathcal{L}'$ is a refinement of some τ-interpretation of $\mathcal{L}$ (i.e., there is a l-deductive system $\mathcal{L}^0$ which τ-interprets $\mathcal{L}$ and $\mathcal{L}^0 \rightsquigarrow \mathcal{L}'$).*

Proof. Suppose $\mathcal{L} \rightharpoonup_\tau \mathcal{L}'$. Then, by Theorem 4.7, $\mathrm{Mod}(\mathcal{L}')$ is a subclass of the class of τ-models of $\mathcal{L}$, *i.e.*, $\mathrm{Mod}(\mathcal{L}') \subseteq \mathrm{Mod}(\mathcal{L}^\tau)$. Therefore, by Theorem 5.3, $\mathcal{L}^\tau \rightsquigarrow \mathcal{L}'$. So, condition (ii) holds for $\mathcal{L}^0 = \mathcal{L}^\tau$.

Suppose now there is a l-deductive system $\mathcal{L}^0$ which τ-interprets $\mathcal{L}$ and $\mathcal{L}^0 \rightsquigarrow \mathcal{L}'$. Let $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$. Then

$$\Gamma \vdash_{\mathcal{L}} \bar{\varphi} \Leftrightarrow \tau(\Gamma) \vdash_{\mathcal{L}^0} \tau(\bar{\varphi}) \Rightarrow \tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi}).$$

The equivalence holds since τ interprets $\mathcal{L}$ in $\mathcal{L}^0$. The implication holds since $\mathcal{L}^\tau \rightsquigarrow \mathcal{L}'$. Therefore, $\mathcal{L} \rightharpoonup_\tau \mathcal{L}'$. □

Example 5.8. Suppose a requirements specification is provided in CPC, but an implementation is sought in which the system properties are expected to be shown in a constructive way resorting, for example, to a theorem prover. This entails the need for refactoring the specification to some variant of intuitionist logic. Based on Theorem 5.7 we have CPC $\rightharpoonup_\tau$ HA $\rightharpoonup_\rho$ IPC, with $\tau(p) = \{\langle \neg\neg p, \top\rangle\}$ and get $\rho(\langle p, q\rangle) = \{p \to q, q \to p\}$ doing the job. ◇

The discussion concerning the composition of refinements by interpretation is not straightforward. For *vertical composition* one gets, similarly to what happens in the equational case,

Theorem 5.9. *Let $\mathcal{L}$, $\mathcal{L}'$ and $\mathcal{L}''$ be k, l and m-deductive systems over Σ, Σ' and Σ'' respectively. Let τ be a (k,l)-translation from Σ to Σ' and ρ a (l,m)-translation from Σ' to Σ''. Suppose that $\mathcal{L} \multimap_\tau \mathcal{L}'$, $\mathcal{L}' \multimap_\rho \mathcal{L}''$ and ρ interprets $\mathcal{L}^\tau$. Then $\mathcal{L} \multimap_{\rho\circ\tau} \mathcal{L}''$*

Proof. Directly from the fact that $\mathcal{L} \multimap_\tau \mathcal{L}'$ and $\mathcal{L}' \multimap_\rho \mathcal{L}''$ we have that $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$ implies $\rho(\tau(\Gamma)) \vdash_{\mathcal{L}''} \rho(\tau(\bar{\varphi}))$ for any $\Gamma \cup \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$.

On the other hand, by hypothesis, for any $\Gamma, \{\bar{\varphi}\} \subseteq \mathrm{Fm}^k(\Sigma)$,

$$\Gamma \vdash_{\mathcal{L}} \bar{\varphi} \Leftrightarrow \tau(\Gamma) \vdash_{\mathcal{L}^\tau} \tau(\bar{\varphi}) \Leftrightarrow \rho(\tau(\Gamma)) \vdash_{(\mathcal{L}^\tau)^\rho} \rho(\tau(\bar{\varphi}))$$

and hence, $\rho \circ \tau$ interprets $\mathcal{L}$. Therefore, $\mathcal{L} \multimap_{\rho\circ\tau} \mathcal{L}''$. □

On the other hand, *horizontal composition* of refinements via interpretation is still a topic of current research, which leads us to the conclusions of this paper.

6. Concluding

6.1. **Related work.** The idea of relaxing what counts as a valid refinement of a specification by replacing *signature morphisms* by *logical interpretations* is, to the best of our knowledge, new. This piece of research was directly inspired by the first author's work on algebraic logic, where the notion of *interpretation* plays a fundamental role (see, *e.g.*, [BP89, BP01, BR03, Cze01]) and occurs in different variants. Rather than reviewing exhaustively this area, we shall concentrate in what appears to be the closest approach, in the literature, to the notion of an interpretation proposed in the paper — that of *conservative translation* intensively studied by Feitosa and D'Ottaviano [Fei97, FD01]. Recall that a conservative translation is a map between deductive system which reflects and preserves logical consequence. It corresponds thus to an interpretation arising from a functional translation with $k = l = 1$, *i.e.*, between sentential languages.

The conjunction property, as characterised in the following definition, allows us to add to the fact that all conservative translations are interpretations (insofar functions are particular cases of multi-functions), its converse, although in a restricted form. Similar properties, also concerning other connectives, have been studied in the framework of the theory of institutions (see [Tar85]).

Definition 6.1. A k-deductive system $\mathcal{L}$ over Σ has the *conjunction property* if, for any $\{\bar{\varphi}_i | i \in \mathrm{I}\} \subseteq \mathrm{Fm}^k(\Sigma)$, for I finite, there exists a $\bar{\xi} \in \mathrm{Fm}^k(\Sigma)$ such that $\{\bar{\varphi}_i | i \in \mathrm{I}\} \dashv\vdash_{\mathcal{L}} \bar{\xi}$. In this case, we denote $\bar{\xi}$ by $\bigwedge_{\mathcal{L}}\{\bar{\varphi}_i | i \in \mathrm{I}\}$.

In the presence of this property, we define the *associated function* of a translation τ between two deductive systems over Σ and Σ' as follows

$$\begin{array}{rccc} f_\tau : & \mathrm{Fm}^k(\Sigma) & \to & \mathrm{Fm}^l(\Sigma') \\ & \bar{\varphi} & \mapsto & \bigwedge_{\mathcal{L}'} \tau(\bar{\varphi}). \end{array}$$

We may now incorporate in the approach proposed in this paper the important tool given in Lemma 6.3:

Lemma 6.2. *Let τ be a self-translation between two 1-deductive systems $\mathcal{L}$ and $\mathcal{L}'$ over the signature Σ. Then, if $\mathcal{L}'$ has the conjunction property, τ is an interpretation iff its associated function is a conservative translation.*

Proof. First we prove that $f_\tau(\Gamma) \dashv\vdash_{\mathcal{L}'} \tau(\Gamma)$: since for any $\xi \in f_\tau(\Gamma)$ there is a $\gamma \in \Gamma$ such that $\xi = \bigwedge_{\mathcal{L}'} \tau(\gamma)$, we have, by the conjunction property of $\mathcal{L}'$, that $\tau(\gamma) \dashv\vdash_{\mathcal{L}'} \xi$ and, by (ii) of Definition 2.1, that $\tau(\Gamma) \vdash_{\mathcal{L}'} f_\tau(\Gamma)$. Analogously, since for each $\gamma \in \Gamma$ there is a $\xi \in f_\tau(\Gamma)$ such $\xi = \bigwedge_{\mathcal{L}'} \tau(\gamma) \dashv\vdash_{\mathcal{L}'} \tau(\gamma)$, we have, by (ii) of Definition 2.1, that $f_\tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\Gamma)$. Hence, for any interpretation τ and for all $\Gamma, \{\varphi\} \subseteq \mathrm{Fm}^1(\Sigma)$,

$$\Gamma \vdash_{\mathcal{L}} \varphi \Leftrightarrow \tau(\Gamma) \dashv\vdash_{\mathcal{L}'} f_\tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\varphi) \dashv\vdash_{\mathcal{L}'} f_\tau(\varphi),$$

which implies that f_τ is a conservative translation. In a similar way, if f_τ is a conservative translation, τ is an interpretation. □

The connection to conservative translations turns out to be very useful in practice. The following theorem, which builds on results in [FD01], provides a sufficient condition for a translation to be an interpretation.

Theorem 6.3. *Let τ be a (k,l)-translation from Σ to Σ', $\mathcal{L}$ a k-deductive system over Σ and $\mathcal{L}'$ a l-deductive system over Σ'. Suppose that τ is functional and injective. If $\tau(\mathrm{Cn}_{\mathcal{L}}(\Gamma)) = \mathrm{Cn}_{\mathcal{L}'}(\tau(\Gamma))$, for every set of formulas Γ, then τ interprets $\mathcal{L}$ in $\mathcal{L}'$.*

Proof. From the inclusion $\tau(\mathrm{Cn}_{\mathcal{L}}(\Gamma)) \subseteq \mathrm{Cn}_{\mathcal{L}'}(\tau(\Gamma))$ we have that $\Gamma \vdash_{\mathcal{L}} \bar{\varphi} \Rightarrow \tau(\Gamma) \vdash_{\mathcal{L}'} \tau(\bar{\varphi})$. Suppose now that $\tau(\bar{\varphi}) \in \mathrm{Cn}_{\mathcal{L}'}(\tau(\Gamma)) = \tau(\mathrm{Cn}_{\mathcal{L}}(\Gamma))$. Hence there is a $\bar{\psi} \in \mathrm{Cn}_{\mathcal{L}}(\Gamma)$ such that $\tau(\bar{\varphi}) = \tau(\bar{\psi})$. Since τ is injective $\bar{\varphi} = \bar{\psi}$, and so, $\bar{\varphi} \in \mathrm{Cn}_{\mathcal{L}}(\Gamma)$, *i.e.*, $\Gamma \vdash_{\mathcal{L}} \bar{\varphi}$. □

The approach to refinement proposed in this paper, in particular when specialised to 2-dimension deductive systems, should also be related to the extensive work of Maibaum, Sadler and Veloso in the 70's and the 80's, as documented, for example, in [MSV84, MVS85]. The authors resort to interpretations between theories and conservative extensions to define a syntactic notion of refinement according to which a specification SP' refines a specification SP if there is an interpretation of SP' into a conservative extension of SP. It is shown that these refinements can be vertically composed, therefore entailing stepwise development. This notion is, however, somehow restrictive since it requires all maps to be conservative, whereas in program development it is usually enough to guarantee that requirements are preserved by the underlying translation. Moreover, in their approach, the interpretation edge of a refinement diagram needs to satisfy extra properties.

As related work one should also mention [FM93, Vou13] where interpretations between theories are studied in the abstract framework of π-institutions. The first reference is a generalisation of the work of Maibaum and his collaborators, whereas the second one generalises the way algebraic semantics on sentential logics is dealt with in abstract algebraic logic to the abstract setting of π-institutions. Similar developments could arise by considering institutions and their (co-)morphisms [GB92, Dia08, Tar96]. The work of Meseguer [Mes89] on *general logics*, in which a theory of interpretations between logical systems is developed, should also be mentioned.

Our own approach to refinement by interpretation can be placed between these general works and the original contribution of Maibaum. Actually, on the one hand, we deal with general k-deductive systems therefore subsuming all frameworks above which are based on equational or first order logic (*i.e.*, on specific instances of k-deductive systems). On the other hand, however, our results are formulated in terms of a concrete and intuitive notion of a deductive system; their scaling to an abstract, institutional level is still to be done.

6.2. **Conclusions and future work.** The paper introduced a new notion of refinement and started the development of a corresponding theory of *refinement by interpretation*. The results obtained and their applications seem promising, in the sense that a number of useful transformations of (classes of models of) specifications are captured as refinement steps. In order to clarify the scope of our results we should point out that the development in Section 3 can be straightforwardly generalised as to apply to any Horn fragment of a structural logic $\mathcal{L}$ (*i.e.*, a logic whose axioms have the form $\bigwedge H \to c$, where $H \cup \{c\}$ is a subset of the atomic formulas of the logic, with H possibly empty). All one has to do is to represent such a fragment by the natural equivalent deductive system taking the atomic formulas of $\mathcal{L}$ as its set of formulas and a presentation given by the axiomatisation of $\mathcal{L}$.

The generalisation made along sections 4 and 5 turns it relevant to the specification meta-level, *i.e.*, whenever an implementation step requires a change in the underlying logic. This often arises in formal software development with the need for accommodating new requirements (as in Example 5.6) or when a particular theorem prover, embodying a specific logic, is to be used for design validation (as in Example 5.8). Our most recent work [MMB13] is another generalisation effort aiming at reframing this notion of refinement in a categorical setting based on a characterisation of abstract logics as coalgebras for the closure system contravariant functor [Pal02] upon the category Set of sets and functions.

To conclude we would like to remark again the 'semantic' perspective from which this work was developed, as extensively discussed in the Introduction. This entails the need for further research on how refinement by interpretation, which is entirely based on properties of arbitrary deductive systems, can be smoothly combined with concrete specification structuring operations. Preliminary work on this topic is reported in [RMMB11] in which the emphasis is shifted to specifications. Current work in this direction includes the development of a refinement calculus of structured specifications over a π-institution.

As a general remark we would like to stress again that the approach developed in this paper can be applied to any notion of algebraic specification based on any fixed set of specification structuring combinators, further justifying the relevance of the 'semantic' perspective adopted in this paper.

A practical limitation of this approach of reducing the specifications to their class of models is that often such classes have infinitely many models. However in some cases subclasses possessing the same theory can be considered instead. For instance, if the class is a finitely generated quasivariety K [Gor98], and consequently the associated 2-deductive system is finitary with a presentation given by the axiomatisation of the quasivariety, we can replace K by the set of its generators which induces the same deductive system. An example is the class of Boolean Algebras which are generated (actually, as a variety) by the two-element Boolean algebra.

From an application point of view, this 'semantic' approach seems to have its own potentialities which we would like to recall. Actually, in a number of cases it is relevant, and even mandatory, to start the implementation procedure from a set of models that does not come from a structured specification. This can be the case when reusing designs (a recurrent strategy in Engineering) or even to express meta-requirements that cannot be easily accommodated within the classic refinement procedure. Focussing on *classes of models*, on the other hand, makes possible to deal with requirements that cannot be properly formalised in a specification. Note this does not entail any loss of expressivity. The approach proposed in this paper can be tuned to specification refinement in a strict sense: for each

specification, one may recursively compute its denotation (a signature and a class of models) and work directly with it.

In general, we believe that this approach has a real application potential, namely to deal with specifications spanning through different specification logics. Particularly deserving to be considered, but still requiring further investigation, are observational logic [BHK03], hidden logic [Roş00, MP07, Mar07, Mar08] and behavioural logic [Hen97]. In all of these cases the satisfaction of requirements is discussed up to some particular satisfaction relation and their verification is checked with respect to relations obtained by replacing strict equality by its underlying notion of satisfaction. In this context, a semantics based on k-structures paves the way to the unification of all of these approaches. Actually, in all of them, models consist of algebras whose k-structures are of the form $\langle A, \Theta \rangle$, where Θ captures the particular satisfaction relation in each formalism. In particular, the strict models of a (classical) algebraic specification SP consist of algebras A whose k-data structure $\langle A, \Delta_A \rangle$ is a model of $\models_{\mathrm{Mod}(SP)}$.

Naturally, most of the models of software specifications are not admissible choices as implementations. Therefore, the choice of adequate filters along the implementation process becomes a crucial, although not trivial task. This should be driven by the system nature (for example, adopting observational equality to deal with objects with encapsulated data). A similar concern is, moreover, shared by other general approaches to formal development, as, for example, [Hen97] in the context of behavioral logic.

A lot of other questions remain to be answered. One such topic, as mentioned above, concerns *horizontal composition* of refinements by interpretation; *vertical* composition raising no special problems as shown in theorems 3.23 and 5.9. To illustrate the kind of results we are investigating suppose, for example, that τ interprets SP in SP'. The challenge would be to prove that τ also interprets an enrichment of SP by axioms in an appropriate sub-specification of SP'. A closely related issue is the extension of this approach to the level of (structured) specifications. We believe that this can be captured in a somehow standard way, which will be most relevant in studying the interplay between horizontal (*i.e.*, architectural) and vertical (*i.e.*, implementation driven) levels of specification composition. For example, the *union* of two specifications will correspond to the union of the corresponding consequence relations. Actually, a structured specification also defines a class of models and therefore induces a deductive system.

Another topic to explore is the equivalence of algebraic specifications up to logical interpretation. As a starting point, it would be worth to explore the relation $\equiv$ defined as follows: $SP \equiv SP'$ if there are interpretations τ and ρ such that $SP \multimap_\tau SP'$ and $SP' \multimap_\rho SP$. 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 deductive systems [CG05, BP89], which requires interpretations to be *mutually* inverse.

Last but not least, framing refinement by interpretation in the context of recent works on heterogeneous specification, raises interesting questions and opens the opportunity for computer-based support. Actually, classical translations between logics (*e.g.*, modal into first-order or the latter into equational logic) are at the basis of Hets [MML07, Mos05, MML09], the heterogeneous specification framework. To go further in this direction entails the need to regard interpretations from an institutional point of view [Dia08], as some sort of comorphisms, and develop on top of it a calculus of refinements by interpretation.

Acknowledgements

The authors express their gratitude to the anonymous referees who raised a number of pertinent questions entailing a more precise characterisation of the paper's contributions and a clarification of their scope. This work was funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028923 (Nasoni) and the project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690 (CIDMA - UA). The first author also acknowledges the financial assistance by the projects GetFun, reference FP7-PEOPLE-2012-IRSES, and Nociones de Completud, reference FFI2009-09345 (MICINN - Spain). A. Madeira was supported by the FCT within the project NORTE-01-0124-FEDER-000060.

References

[AKKB99] E. Astesiano, H. Kreowski, and B. Krieg-Brückner. *Algebraic Foundations of Systems Specification*. IFIP state-of-the-art reports. Springer, 1999.

[BD74] R. Balbes and P. Dwinger. *Distributive lattices*. Columbia, Missouri: University of Missouri Press, 1974.

[BG80] R. M. Burstall and J. A. Goguen. The semantics of CLEAR, a specification language. In D. Bjørner, editor, *Abstract Software Specifications (1979 Copenhagen Winter School, January 22 - February 2, 1979)*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[BH08] M. Bidoit and R. Hennicker. An algebraic semantics for contract-based software components. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology (AMAST 2008 - Urbana, IL, USA, July 28-31, 2008*, volume 5140 of *Lecture Notes in Computer Science*, pages 216–231. Springer, 2008.

[BHK03] M. Bidoit, R. Hennicker, and A. Kurz. Observational logic, constructor-based logic, and their duality. *Theor. Comput. Sci.*, 298(3):471–510, 2003.

[BM13] S. Babenyshev, M. A. Martins. Deduction-detachment theorem in hidden$k$-logics *Journal of logic and Computation* (doi:10.1093/logcom/ext008)

[Bor02] T. Borzyszkowski. Logical systems for stuctured specifications. *Theor. Comp. Science*, 286:197–245, 2002.

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

[BP01] W. Blok and D. Pigozzi. Abstract algebraic logic and the deduction theorem. 2001. Preprint. Available at http://www.math.iastate.edu/dpigozzi/ papers/aaldedth.pdf.

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

[CCD09] W. A. Carnielli, M. E. Coniglio, and I. M. D'Ottaviano. New dimensions on translations between logics. *Logica Universalis*, 3(1):1–18, 2009.

[CDE+07] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

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

[CHK+11] M. Codescu, F. Horozal, M. Kohlhase, T. Mossakowski, and F. Rabe. Project abstract: Logic atlas and integrator (latin). In James Davenport, William Farmer, Josef Urban, and Florian

Rabe, editors, *Intelligent Computer Mathematics*, volume 6824 of *Lecture Notes in Computer Science*, pages 289–291. Springer Berlin / Heidelberg, 2011.

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

[DD05] B. Dolle and W. 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. IEEE Computer Society, 2005.

[DF98] R. Diaconescu and K. Futatsugi. *CafeOBJ report: the language, proof techniques, and methodologies for object-oriented algebraic specification*. AMAST series in computing. World Scientific, 1998.

[Dia08] R. Diaconescu. *Institution-independent Model Theory*. Series in Universal Logic. Birkhauser, 2008.

[DT11] R. Diaconescu and I. Tutu. On the algebra of structured specifications. *Theor. Comput. Sci.*, 412(28):3145–3174, 2011.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. Springer-Verlag, 1985.

[Fav98] L. Favre. Object oriented reuse through algebraic specifications. In *Technology of Object-Oriented Languages and Systems (TOOLS'98, Melbourne, 23-26 Nov, 1998)*. IEEE Computer Society, 1998.

[FD01] H. A. Feitosa and I. M. 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.

[FJP03] J. M. Font, R. Jansana, and D. Pigozzi. A survey of abstract algebraic logic. *Stud. Log.*, 74(1-2):13–97, 2003.

[FM93] J. Fiadeiro and T. S. Maibaum. Generalising interpretations between theories in the context of (π-) institutions. In *Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods*, pages 126–147, London, UK, 1993. Springer-Verlag.

[FS88] J. Fiadeiro and A. Sernadas. Structuring theories on consequence. In D. Sanella and A. Tarlecki, editors, *Recent Trends in Data Type Specification. Specification of Abstract Data Types (Papers from the Fifth Workshop on Specification of Abstract Datac Types, Gullane, 1987)*, volume 332 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1988.

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

[GH78] J. V. Guttag and J. J. Horning. The algebraic specification of abstract data types. *Acta Informatica*, 10:27–52, 1978.

[Gli29] V. Glivenko. Sur quelques points de la logique de M. Brouwer. *Bulletins de la classe des sciences*, 15(5):183–188, 1929.

[Göd86] K. Gödel. An interpretation of the intuitionistic proposicional calculus (1933). In S. Feferman et al, editor, *Collected works of Kurt Gödel (vol. I)*, pages 301–303. Oxford: Oxford University Press, 1986.

[Gor98] V. A. Gorbunov. *Algebraic Theory of Quasivarieties*. Siberian School of Algebra and Logic. Springer, 1998.

[GTW78] J. Goguen, J. Thatcher, and E. Wagner. An initial algebra approach to the specification, correctness and implementation of abstract data types. In R. Yeh, editor, *Current Trends in Programming Methodology*, pages 80–149. Prentice-Hall International, 1978.

[GTWW77] J. Goguen, J. Thatcher, E. Wagner, and J. Wright. Initial algebra semantics and continuous algebras. *Jour. of the ACM*, 24(1):68–95, January 1977.

[Gut75] J. V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, Dept. of Computer Science, University of Toronto, 1975.

[GWM+96] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Practice*. Cambridge University Press, 1996.

[Hen97] R. Hennicker. Structural specifications with behavioural operators: semantics, proof methods and applications, 1997. Habilitationsschrift.

[Hoa72] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–281, 1972.

[HM13] D. Hofmann and M. A. Martins. On a coalgebraic view on Logic. *Journal of logic and Computation*, 23(5):1097–1106, 2013 (doi: 10.1093/logcom/exs063).

[HRD08] J. Henkel, C. Reichenbach, and A. Diwan. Developing and debugging algebraic specifications for java classes. *ACM Transactions on Software Engineering and Methodology*, 17(3):14–37, 2008.

[Kol77] A. N. Kolmogorov. On the principle of excluded middle (1925). In J. Hei-Jenoort, editor, *From Frege to Gödel: a source book in mathematical logic 1879–1931*, pages 414–437. Cambridge: Harvard University Press, 1977.

[LEW00] J. Loeckx, H.-D. Ehrich, and M. Wolf. Algebraic specification of abstract data types. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Theoretical Computer Science, Volume 5*. Oxford University Press, 2000.

[LZ74] B. Liskov and S. N. Zilles. Programming with abstract data types. In *Proceedings of ACM SIGPLAN Symposium on Very High Level Programming Languages*. SIGPLAN Notices, 9 (4), 1974.

[Mad08] A. Madeira. Observational refinement process. *Electr. Notes Theor. Comput. Sci.*, 214:103–129, 2008.

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

[Mar07] M. A. Martins. Closure properties for the class of behavioral models. *Theor. Comput. Sci.*, 379(1-2):53–83, 2007.

[Mar08] M. A. Martins. On the behavioral equivalence between k-data structures. *The Computer Journal*, 51(2):181–191, 2008.

[MDT09] T. Mossakowski, R. Diaconescu, and A. Tarlecki. What is a logic translation. *Logica Universalis*, 3(1):95–124, 2009.

[Mes89] J. Meseguer. General logics. In H. D. Ebbinghaus, J. Fernandez-Prida, M. Garrido, D. Lascar, and M. Rodriguez-Artalejo, editors, *Logic Colloqium'87*, Studies in Logic and the Foundations of Mathematics (volume 129), pages 275–330. Elsevier, 1989.

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

[MMB09a] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation in a general setting. *Electron. Notes Theor. Comput. Sci.*, 259:105–121, 2009.

[MMB09b] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement via interpretation. In D. V. Hung and P. Krishnan, editors, *Seventh IEEE International Conference on Software Engineering and Formal Methods (SEFM 2009, Hanoi, Vietnam, 23-27 November 2009)*, pages 250–259. IEEE Computer Society, 2009.

[MMB13] M. A. Martins, A. Madeira, and L. S. Barbosa. A coalgebraic perspective on logical interpretations. *Studia Logica - Special Issue on Abstract Algebraic Logic*, 101(4):783–825, 2013.

[MML07] T. Mossakowski, C. Maeder, and K. Lüttich. The heterogeneous tool set, Hets. In O. Grumberg and M. Huth, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007 - Braga, Portugal, March 24 - April 1, 2007)*, volume 4424 of *Lecture Notes in Computer Science*, pages 519–522. Springer, 2007.

[MML09] T. Mossakowski, C. Maeder, and K. Lüttich. Hets user guide (version 0.85), 2009.

[Mos05] T. Mossakowski. Heterogeneous specification and the heterogeneous tool set, 2005. Habilitation thesis.

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

[MSV84] T. S. Maibaum, M. R. Sadler, and P. A. Veloso. Logical specification and implementation. In J. Mathai and R. K. Shyamasundar, editors, *Foundations of Software Technology and Theoretical Computer Science (FSTTCS, Bangalore, India, Dec.13-15, 1984)*, volume 4424 of *Lecture Notes in Computer Science*, pages 13–30, London, UK, 1984. Springer.

[MVS85] T. S. E. Maibaum, Paulo A. S. Veloso, and M. R. Sadler. A theory of abstract data types for program development: Bridging the gap? In Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James W. Thatcher, editors, *Mathematical Foundations of Software Development*

(TAPSOFT, Berlin, Germany, March 25-29, 1985), volume 186 of *Lecture Notes in Computer Science*, pages 214–230. Springer, 1985.

[Pal02] A. Palmigiano. Abstract logics as dialgebras. *Electr. Notes Theor. Comput. Sci.*, 65(1), 2002.

[Par72] D. Parnas. Information distribution aspects of design methodology. In *Information Processing '72*, pages 339–344. North-Holland, 1972.

[Pig91] D. Pigozzi. Equality-test and if-then-else algebras: Axiomatization and specification. *SIAM J. Comput.*, 20(4):766–805, 1991.

[PM68] D. Prawitz and P.-E. Malmnäs. A survey of some connections between classical, intuitionistic and minimal logic. In *Contributions to Mathematical Logic: Proc. Logic Colloq. (Hannover 1966)*, pages 215–229. North-Holland, 1968.

[Rab08] F. Rabe. *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen, 2008.

[RMMB11] C. J. Rodrigues, M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation in π-institutions. In John Derrick, Eerke A. Boiten, and Steve Reeves, editors, *Proceedings 15th International Refinement Workshop*, volume 55 of *EPTCS*, pages 53–64, 2011.

[Roş00] G. Roşu. *Hidden Logic*. PhD thesis, University of California, San Diego, 2000.

[San01] D. Sannella. The common framework initiative for algebraic specification and development of software: Recent progress. In M. Cerioli and G. Reggio, editors, *Recent Trends in Algebraic Development Techniques (Revised Selected Papers of WADT 2001,Genova, Italy, April 1-3, 2001)*, volume 2267 of *Lecture Notes in Computer Science*, pages 328–344. Springer, 2001.

[SDS99] J. Da Silva, I. M. D'Ottaviano, and A. M. Sette. Translations between logics. In *Models, algebras, and proofs: Selected papers of the X Latin American Symposium on Mathematical Logic, (Bogotá, 1995)*, pages 435–448. Lect. Notes Pure Appl. Math. (203), 1999.

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

[ST06] D. Sannella and A. Tarlecki. Horizontal composability revisited. In K. Futatsugi, J.-P. Jouannaud, and J. Meseguer, editors, *Algebra, Meaning, and Computation, Essays Dedicated to Joseph A. Goguen on the Occasion of His 65th Birthday*, volume 4060 of *Lecture Notes in Computer Science*, pages 296–316. Springer, 2006.

[ST11] D. Sannella and A. Tarlecki. *Foundations of algebraic specification and formal software development*. Springer Verlag, 2011.

[Tar56] A. Tarski. *Logic, semantics, meta-mathematics. Papers from 1923 to 1938. Translated by J. H. Woodger*. Oxford: Clarendon Press; London: G. Cumberlege XIV, 471 p. , 1956.

[Tar85] A. Tarlecki. On the existence of free models in abstract algebraic institutuons. *Theor. Comp. Science*, 37:269–304, 1985.

[Tar96] A. Tarlecki. Moving between logical systems. In Magne Haveraaen, Olaf Owe, and Ole-Johan Dahl, editors, *Recent Trends in Data Type Specification (Selected Papers of 11th Workshop on Specification of Abstract Data Types Joint with the 8th COMPASS Workshop, Oslo, Norway, September 19-23, 1995)*, volume 1130 of *Lecture Notes in Computer Science*, pages 478–502. Springer, 1996.

[Vou03] G. Voutsadakis. Categorical abstract algebraic logic: Equivalent institutions. *Studia Logica*, 74:275–311, 2003.

[Vou13] G. Voutsadakis. Categorical abstract algebraic logic: Algebraic semantics for π-institutions. *Math. Log. Q.*, 59(3):177–200, 2013.

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

[Wój88] R. Wójcicki. *Theory of logical caculi. Basic theory of consequence operations*. Synthese Library, 199. Dordrecht etc.: Kluwer Academic Publishers., 1988.

[YKZZ08] B. Yu, L. Kong, Y. Zhang, and H. Zhu. Testing java components based on algebraic specifications. In *First International Conference on Software Testing, Verification, and Validation, ICST 2008*, pages 190–199. IEEE Computer Society, 2008.