M. A. Martins
A. Madeira
L. S. Barbosa

A coalgebraic perspective on logical interpretations

Abstract. In Computer Science stepwise refinement of algebraic specifications is a well-known formal methodology for rigorous program development. This paper illustrates how techniques from Algebraic Logic, in particular that of *interpretation*, understood as a multifunction that preserves and reflects logical consequence, capture a number of relevant transformations in the context of software design, reuse, and adaptation, difficult to deal with in classical approaches. Examples include *data encapsulation* and the *decomposition of operations into atomic transactions*. But if interpretations open such a new research avenue in program refinement, (conceptual) tools are needed to reason about them. In this line, the paper's main contribution is a study of the correspondence between logical interpretations and morphisms of a particular kind of coalgebras. This opens way to the use of coalgebraic constructions, such as simulation and bisimulation, in the study of interpretations between (abstract) logics.

Keywords: abstract logic, interpretation, coalgebra, program refinement.

1. Introduction and overview

1.1. Motivation and objectives

Defining translation maps to interrelate logics and to connect their properties, has been common practice since the beginning of last century. These *translations* were investigated as part of an ambitious programme addressing tools to handle the multiplicity of logics.

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, though, a translation is regarded as a map between the sets of formulas of different logics such that the image of a theorem is still a theorem. They were used at first to understand the relationship between classical and constructive logics. Soon, however, their scope of applications broadened. 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 [22], Glivenko [18] and Gentzen [19] involving classical, intuitionistic and modal logics.

Presented by **Name of Editor**; *Received* December 1, 2005

To the best of our knowledge, the first known general definition for the concept of translation between logical systems is due to Prawitz and Malmnäs [41]. More recently, Wójcicki [50] presented a systematic study of translations between logics, focussing on inter-relations between sentential logics. And the quest goes on (cf. [35, 9, 10]). At the turn of the century, Silva, D'Ottaviano and Sette [46] proposed a general definition of translations between logics as maps preserving consequence relations. Then Feitosa and D'Ottaviano studied intensively the subclass of translations that preserve and reflect consequence relations, coining the name *conservative translations* [15].

The notion of translation proposed in the present paper generalizes conservative translations by allowing maps to be *multifunctions* but still preserving and reflecting consequence. Such a general notion has been used in abstract algebraic logic, under the name of *interpretation*, in the study of equivalent algebraic semantics [6]. The latter intends to generalize the duality between classic propositional logic and the class of Boolean algebras to other logics. This generalization is captured by the notion of algebraizable logic. A logic $\mathcal{L} = \langle \Sigma, \vdash \rangle$ is said to be *algebraizable* whenever there exists a class K of algebras such that the equational consequence relation $\models_K$ is equivalent to $\vdash$. Such an equivalence was originally defined by means of mutually inverse interpretations τ and ρ commuting with arbitrary substitutions. Since then, this link between logic and universal algebra has been successfully explored. In particular, for an algebraizable logic $\mathcal{L}$, logical properties of $\mathcal{L}$ can be related to algebraic properties of its equivalent algebraic semantics. This kind of results are often called as *bridge theorems*, of which many examples exist. A well-known one states that an algebraizable logic has the Craig's interpolation property if and only if the class of the algebraic reducts of its reduced matrix models has the amalgamation property [11].

Our interest in logical interpretations was boosted by their suitability to capture difficult problems in quite a different setting, that of Computer Science, as explained below.

Actually, in Computer Science, translations between logics (used as program specifications) are usually witnessed by signature morphisms. Briefly, a signature morphism is a map between sorts and a family of functions, one for each operator in the signature, respecting the sort translation. It extends to formulas in a natural way and, consequently, provides a handy way to relate specifications. It is easy to see, by the Satisfaction Lemma, that a signature morphism always preserves consequence; but, in general, does not reflect it. Hence, it is appropriate to consider translations between

logics as conservative signature morphisms, i.e., signature morphisms that preserve and reflect consequence. Sometimes, it is even worth considering a more general notion of signature morphism mapping an operation symbol to a derived operation symbol in the target signature.

In a recent series of papers [29, 30] we introduced an alternative approach to refinement of specifications in which signature morphisms are replaced by *logic interpretations*. Introducing logic interpretations proved effective in capturing a number of transformations difficult to deal with in classical terms. Such is the case, for example, of transformations associated to *data encapsulation* and to the *decomposition of operations into atomic transactions*. The use of logic interpretations may also be relevant in the context of new, emerging computing paradigms which entail the need for more flexible approaches to what is taken as a valid transformation of specifications (see, for example, [4]).

1.2. Overview

If interpretations look promising for the line of research we described, which seeks applications to program refinement, (conceptual) tools are needed to reason about them. Such is the purpose of the present paper.

More precisely we set a correspondence between logical interpretations and morphisms of a particular kind of coalgebras, so that usual coalgebraic constructions, such as simulations and bisimulations, can be used to explore interpretations between (abstract) logics. On intuitive grounds, the idea seems promising: coalgebra morphisms do indeed *preserve* and *reflect* the structure of the underlying coalgebra, just as interpretations *preserve* and *reflect* logical consequence.

The programme itself is not new: connections between the theory of consequence operators, as defined in [50], and coalgebra were first introduced in [37]. Our contribution extends this work from strict logical morphisms to interpretations, which, as *multifunctions*, are much more flexible, yet less straightforward to deal with.

After a brief review of multifunctions and coalgebras in section 2, section 3 introduces logical interpretations and studies some of their properties. Section 4 shows how interpretations can be regarded as coalgebra morphisms for a specific endofunctor in the category of families of sets and exemplifies some advantages one may take from such a relationship. Applications to program refinement are discussed in section 5. Section 6 concludes and enumerates a few topics for future research.

2. Preliminaries

Since the purpose of the present paper is to frame *logical interpretations* as morphisms of a specific class of *coalgebras*, it seems appropriate to briefly introduce, in this section, the basic underlying constructions: *multifunctions* and *coalgebra morphisms*. We start this background section, however, by revisiting the notion of a *binary relation*, which subsumes both and whose calculus provides an agile tool for proofs. A number of results on multifunctions, required later in the paper, are proved here.

Relations. Let $R : A \longrightarrow B$ denote a binary relation on sets A (source) and B (target). We write aRb to mean that the pair $\langle a, b\rangle$ is in R. The underlying partial order on relations with the same source and target sets is written $R \subseteq S$, meaning that S is either more defined or less deterministic than R. Relations can be combined by three basic operators: composition ($R \cdot S$), converse (R°) and meet ($R \cap S$). Meet corresponds to set-theoretical intersection and composition is defined as usual: for $R : A \longrightarrow B$, $S : B \longrightarrow C$, $a(S \cdot R)c$ holds whenever there exists some mediating $b \in B$ such that $aRb \wedge bSc$. Alternative notation $R; S$, expressing composition diagramatically, is often used in the literature on binary relations (*e.g.*, [17, 25]). R° is the relation such that $aR^\circ b$ iff bRa holds.

The calculus of binary relations was introduced in 1860 by Augustus de Morgan and was further developed in the second half of the nineteenth-century by Charles Sanders Peirce and Ernst Schröder. In 1940, Alfred Tarski proposed an elegant axiomatization of the calculus [48] which led to the creation of relation algebras and shaped the subject as we know it today [24, 40]. Since the 1960s, relations have been used in a categorical setting [17] and applied to various areas of computer science [5]. Such a setting not only paves the way to generalization, but also helps in structuring the calculus as detailed below.

The category Rel of sets and relations is the archetypal example of an allegory [17]: composition and the identity relation id_A, for every set A, provide the categorical structure; converse and composition are monotonic with respect to $\subseteq$; meet verifies the universal property[1]:

$$T \subseteq R \cap S \;\Leftrightarrow\; (T \subseteq R) \wedge (T \subseteq S) \tag{1}$$

[1] All the laws introduced in the paper hold for arguments of suitable, compatible types, though type information is left implicit whenever it can be inferred from the context.

and, finally, the following laws hold for converse

$$(R^\circ)^\circ \;=\; R \tag{2}$$
$$(S \cdot R)^\circ \;=\; R^\circ \cdot S^\circ \tag{3}$$
$$(S \cdot R) \cap T \;=\; S \cdot (R \cap (S^\circ \cdot T)) \tag{4}$$

Additional structure makes Rel a rich mathematical universe for specifications. A first observation is that its hom-sets, *i.e.*, the collections of relations with the same source and target, form a bounded distributive lattice with set-theoretical union $\cup$ as join and $\top = A \times B$ as the largest relation of type $A \to B$. Its dual $\bot$, the smallest such relation, is of course the empty relation. Join satisfies a universal property dual to (1):

$$R \cup S \subseteq T \;\Leftrightarrow\; (R \subseteq T) \wedge (S \subseteq T) \tag{5}$$

and distributes over composition and meet:

$$(R \cup S) \cdot T \;=\; R \cdot T \;\cup\; S \cdot T \tag{6}$$
$$Q \cap (R \cup S) \;=\; (Q \cap R) \cup (Q \cup S) \tag{7}$$

A relation $R : A \longrightarrow B$ is a function if it is both *simple* (or functional) *i.e.*, $R \cdot R^\circ \subseteq id_B$, and *entire* (or total), *i.e.*, $id_A \subseteq R^\circ \cdot R$. In the sequel functions will be denoted by lowercase letters. Juxtaposition will be used for function application, writing $f\, a = b$ to mean $\langle a, b\rangle \in f$. The interplay of functions and relations is a rich part of the binary relation calculus. In particular, functions can be shunted from one side of an inequation to the other:

$$f \cdot R \subseteq S \;\Leftrightarrow\; R \subseteq f^\circ \cdot S \tag{8}$$
$$R \cdot f^\circ \subseteq S \;\Leftrightarrow\; R \subseteq S \cdot f \tag{9}$$

We prove (8) as an illustration of the calculus:

$$\begin{array}{ll} & f \cdot R \subseteq S \\ \Rightarrow & \quad \{ \text{ monotonicity of composition } \} \\ & f^\circ \cdot f \cdot R \subseteq f^\circ \cdot S \\ \Rightarrow & \quad \{ \text{ } f \text{ entire and monotonicity } \} \\ & R \subseteq f^\circ \cdot S \\ \Rightarrow & \quad \{ \text{ monotonicity of composition } \} \end{array}$$

$$
\begin{array}{ll}
& f \cdot R \subseteq f \cdot f^\circ \cdot S \\
\Rightarrow & \qquad \{ \ f \text{ simple and monotonicity } \} \\
& f \cdot R \subseteq S
\end{array}
$$

Relations with a common target, say $R : A \longrightarrow C$ and $S : B \longrightarrow C$, can be *divided* yelding $(R/S) : A \longrightarrow B$ which verifies the following universal property

$$T \subseteq (R/S) \ \Leftrightarrow \ S \cdot T \subseteq R \tag{10}$$

Equivalence (10) defines R/S as the greatest relation whose composition with S is at most R. Going pointwise, condition $S \cdot T \subseteq R$ corresponds to predicate

$$\forall_{a \in A, c \in C} \ . \ (\exists_{b \in B} \ . \ aTb \wedge bSc) \Rightarrow aRc \tag{11}$$

Fixing points a and b and choosing the greatest T such that (11) holds, entails the usual pointwise definition of relational division: $a(R/S)b$ iff $\forall_{c \in C} . \, bSc \Rightarrow aRc$. Similarly a dual division operator can be defined for relations with a common source. Both are particularly useful in abstracting, *i.e.*, converting to relational form, formulas involving universal quantifiers. The following property will be required later:

$$(R \cdot f)/S \ = \ (R/S) \cdot f \tag{12}$$

which can be proved by indirect equality as follows

$$
\begin{array}{ll}
& X \subseteq (R \cdot f)/S \\
\Leftrightarrow & \qquad \{ \text{ division universal property (10) } \} \\
& S \cdot X \subseteq R \cdot f \\
\Leftrightarrow & \qquad \{ \text{ shunting (9) } \} \\
& S \cdot X \cdot f^\circ \subseteq R \\
\Leftrightarrow & \qquad \{ \text{ division universal property (10) } \} \\
& X \cdot f^\circ \subseteq R/S \\
\Leftrightarrow & \qquad \{ \text{ shunting (9) } \} \\
& X \subseteq (R/S) \cdot f
\end{array}
$$

Multifunctions. Multifunctions, as suggested by the symbol used in this paper for their declaration, $m : A \prec\!\!\!\leq B$, are set-valued functions, *i.e.*, $m :$

$A \longrightarrow \mathscr{P}B$, where $\mathscr{P}B$ is the power set of B. Given two multifunctions $m : A \prec\!\!\triangleleft B$ and $n : B \prec\!\!\triangleleft C$, their composition is defined as follows:

$$n \bullet m \;=\; \bigcup \cdot \mathscr{P}n \cdot m \tag{13}$$

which is depicted in Set, the usual category of sets and functions, as follows

$$A \xrightarrow{\;m\;} \mathscr{P}B \xrightarrow{\;\mathscr{P}n\;} \mathscr{P}(\mathscr{P}C) \xrightarrow{\;\bigcup\;} \mathscr{P}C$$

As the definition of their composition may suggest, multifunctions are the arrows of the Kleisli category for the power set monad [21]. Recall that, for each set A, multiplication for this monad is distributed set union, *i.e.*, multifunction $\bigcup_A : \mathscr{P}(\mathscr{P}A) \prec\!\!\triangleleft A$, whereas unit, $\eta_A : A \prec\!\!\triangleleft A$, assigns to each $a \in A$ the singleton set $\{a\}$. Subscripts will be dropped whenever clear form the context. Both $\bigcup$ and η are natural transformations, *i.e.*, for each function $f : A \longrightarrow B$,

$$\mathscr{P}f \cdot \bigcup \;=\; \bigcup \cdot \mathscr{P}(\mathscr{P}f) \tag{14}$$

$$\mathscr{P}f \cdot \eta \;= \eta \cdot f \tag{15}$$

and satisfy the usual monad laws:

$$\bigcup \cdot \bigcup \;=\; \bigcup \cdot \mathscr{P} \bigcup \tag{16}$$

$$\bigcup \cdot \eta \;=\; \bigcup \cdot \mathscr{P}\eta \;=\; id \tag{17}$$

Composition $\eta \cdot f$, which turns f into a multifunction, will be abbreviated to ηf along the paper. The following result, for any $f : A \longrightarrow B$ and $g : B \longrightarrow C$, will be used later:

$$\eta g \bullet \eta f \;=\; \eta(g \cdot f) \tag{18}$$

Its proof illustrates the use of the monad laws above:

$$\begin{array}{ll} & \eta g \bullet \eta f \\ = & \qquad \{ \text{ definitions of } \bullet \text{ and } \eta f;\ \mathscr{P} \text{ is a functor } \} \\ & \bigcup \cdot \mathscr{P}\eta \cdot \mathscr{P}g \cdot \eta \cdot f \\ = & \qquad \{ \text{ (15) } \} \\ & \bigcup \cdot \mathscr{P}\eta \cdot \eta \cdot g \cdot f \\ = & \qquad \{ \text{ (17) and definition of } \eta f \ \} \\ & \eta(g \cdot f) \end{array}$$

Replacing ηf above by an arbitrary multifunction $m : B \prec\!\!\!\!\cdot\, C$ yields

$$m \bullet \eta f \;=\; m \cdot f \tag{19}$$

because

$$m \bullet \eta f = \bigcup \cdot \mathscr{P} m \cdot \eta \cdot f = \bigcup \cdot \eta \cdot m \cdot f = m \cdot f$$

A well-known bijective correspondence between such functions and binary relations leads to another, somehow more basic, characterization of multifunctions as relations $M : A \longrightarrow B$, defined by aMb iff $b \in m\, a$. The bijection is established by the *power transpose* operator Λ from the category Rel of sets and binary relations to the category Set of sets and functions. When applied to a relation $M : A \longrightarrow B$ it yields a multifunction $\Lambda M : A \prec\!\!\!\!\cdot\, B$ defined by

$$(\Lambda M)\, a \;=\; \{b \mid aMb\}$$

This definition can be restated as a universal property

$$m = \Lambda M \;\Leftrightarrow\; \ni_B \cdot m = M \tag{20}$$

where $\ni_B \colon \mathscr{P} B \longrightarrow B$ is the (converse) membership relation. Again the subscript in $\ni_B$ will be omitted whenever this does not create ambiguities. Also note that the (converse) subset relation is obtained by division, as can be easily checked from the definition of $\ni$:

$$\supseteq \;=\; \ni \,/\, \ni \tag{21}$$

Equivalence (20) establishes Λ and $(\ni \cdot)$ as both upper and lower adjoints of each other. A direct consequence of this fact is that Λ distributes over union and intersection. Similarly,

$$R \subseteq S \;\Rightarrow\; \Lambda(R) \mathbin{\dot{\subseteq}} \Lambda(S) \tag{22}$$

where the pointwise lifting of a partial order $\leq$ to functions is represented by $\dot{\leq}$, defined as $f \mathbin{\dot{\leq}} g \;\Leftrightarrow\; f \subseteq\; (\leq \cdot g)$. Other useful properties are also derived in one step from (20):

$$\ni \;\cdot \Lambda R \;=\; R \tag{23}$$

$$\Lambda(R \cdot f) \;=\; \Lambda R \cdot f \tag{24}$$

$$\Lambda \ni \;=\; id \tag{25}$$

Both multiplication and unit of the power set monad, mentioned above, can be characterized through the power transpose and (converse) membership:

$$\bigcup = \Lambda(\ni \cdot \ni) \tag{26}$$

$$\eta = \Lambda(id) \tag{27}$$

And similarly for the Kleisli composition

$$n \bullet m = \Lambda(\ni \cdot n \cdot \ni) \cdot m \tag{28}$$

because

$$\begin{array}{ll} & n \bullet m \\ = & \quad \{ \text{ definition (13) } \} \\ & \bigcup \cdot \mathscr{P} n \cdot m \\ = & \quad \{ \text{ (26) } \} \\ & \Lambda(\ni \cdot \ni) \cdot \mathscr{P} n \cdot m \\ = & \quad \{ \text{ (24) } \} \\ & \Lambda(\ni \cdot \ni \cdot \mathscr{P} n) \cdot m \\ = & \quad \{ \text{ } \ni \text{ natural: } \ni \cdot \mathscr{P} f = f \cdot \ni \} \\ & \Lambda(\ni \cdot n \cdot \ni) \cdot m \end{array}$$

Equivalence (20) can also be regarded as establishing an isomorphism of order-enriched categories between Rel and $\mathsf{Kl}(\mathscr{P})$. From the former to the latter the correspondence is established by functor $\Lambda : \mathsf{Rel} \longrightarrow \mathsf{Kl}(\mathscr{P})$. Conversely, the inverse functor is defined by the assignment $m \mapsto \ni_B \cdot m$.

The *existential image* functor, $\mathscr{E}$, from the category Rel of binary relations to its sub-category of functions, maps a relation $R : A \longrightarrow B$ to a function $\mathscr{E}R : \mathscr{P}A \longrightarrow \mathscr{P}B$ given by

$$\mathscr{E}R = \Lambda(R \cdot \ni) \tag{29}$$

or, put into a pointwise style, $\mathscr{E}R\,X = \{b \mid \exists_{a \in A} \;\; aRb \wedge a \in X\}$. An "absorption" result for $\mathscr{E}R$

$$\mathscr{E}R \cdot \Lambda S \;\; = \;\; \Lambda(R \cdot S) \tag{30}$$

is proven as follows:

$$\begin{array}{ll} & \mathscr{E}R \cdot \Lambda S \;=\; \Lambda(R \cdot S) \\ \Leftrightarrow & \qquad \{\ (29)\ \} \\ & \Lambda(R \cdot \ni) \cdot \Lambda S \;=\; \Lambda(R \cdot S) \\ \Leftrightarrow & \qquad \{\ (20)\ \} \\ & \ni \cdot \Lambda(R \cdot \ni) \cdot \Lambda S \;=\; R \cdot S \\ \Leftrightarrow & \qquad \{\ (23) \text{ twice } \} \\ & R \cdot S \;=\; R \cdot S \end{array}$$

For functions, $\mathscr{E}f$ coincides with $\mathscr{P}f$, but for a multifunction m, what is usually called its direct image (and often represented as $m[X]$) is the existential image of the corresponding relation $(\ni \cdot m)$, which here and in what follows we denote by M, *i.e.*,

$$m^* X \;=\; \mathscr{E}M\, X \;=\; \{b \mid \exists_{a \in A}\ b \in m\, a \wedge a \in X\} \;=\; \bigcup \cdot \mathscr{P}m\, X$$

which the reader may recognize, through notation m^*, as the Kleisli arrow for the powerset monad [42].

The inverse image of a function f is simply $\mathscr{E}f^\circ$, where f° is the converse of f regarded as a binary relation. For multifunctions, however, $\mathscr{E}M^\circ\, Y = \{a \mid m\, a \cap Y \neq \emptyset\}$, whereas the usual definition is $m^{-1}\, Y = \{a \mid m\, a \subseteq Y\}$, *i.e.*,

$$m^{-1} \;=\; \Lambda(m^\circ \cdot \supseteq) \tag{31}$$

For $m : A \prec\!\!\!\!\circ\; B$, $m^* : \mathscr{P}A \longrightarrow \mathscr{P}B$ and $m^{-1} : \mathscr{P}B \longrightarrow \mathscr{P}A$ compose and their composition satisfies

$$id \;\dot{\subseteq}\; m^{-1} \cdot m^* \;\text{ and }\; m^* \cdot m^{-1} \;\dot{\subseteq}\; id$$

which corresponds to the similar general result $id \;\dot{\subseteq}\; \mathscr{E}R^\circ \cdot \mathscr{E}R$ and $\mathscr{E}R \cdot \mathscr{E}R^\circ \;\dot{\subseteq}\; id$, for an arbitrary relation R. A calculational proof of the second inequality follows:

$$\begin{array}{ll} & m^* \cdot m^{-1} \\ = & \qquad \{\ \text{definition of } m^* \text{ and } m^{-1}\ \} \\ & \mathscr{E}M \cdot \Lambda(m^\circ \cdot \supseteq) \\ = & \qquad \{\ (30)\ \} \end{array}$$

$$
\begin{array}{ll}
 & \Lambda(M \cdot m^{\circ} \cdot \supseteq) \\
= & \quad \{ \text{ definition of } M \ \} \\
 & \Lambda(\ni \cdot m \cdot m^{\circ} \cdot \supseteq) \\
\dot{\subseteq} & \quad \{ \text{ as a function } m \text{ satisfies } m \cdot m^{\circ} \subseteq id \text{ and (22) } \} \\
 & \Lambda(\ni \cdot \supseteq) \\
= & \quad \{ \ (\ni \cdot \supseteq) = \ni \text{ and (25) } \} \\
 & id
\end{array}
$$

Unfolding the definition of inverse image of a multifunction yields

$$m\,x \subseteq Y \ \Leftrightarrow \ x \in m^{-1}\,X \tag{32}$$

which lifts to

$$m^{*}X \subseteq Y \ \Leftrightarrow \ X \subseteq m^{-1}\,Y \tag{33}$$

Other properties of m^{*} and m^{-1} will be required later in the paper. The universal property of $\cap$ yields for free $m^{*} \cdot \cap \,\dot{\subseteq}\, \cap \cdot (m^{*} \times m^{*})$ and, similarly, for m^{-1}. Actually,

$$
\begin{array}{ll}
 & m^{*}\,(X \cap Y) \subseteq m^{*}\,X \ \cap \ m^{*}\,Y \\
\Leftrightarrow & \quad \{ \ (1) \ \} \\
 & m^{*}\,(X \cap Y) \subseteq m^{*}\,X \ \wedge \ m^{*}\,(X \cap Y) \subseteq m^{*}\,Y \\
\Leftarrow & \quad \{ \ m^{*} \text{ is monotonic } \} \\
 & X \cap Y \subseteq X \ \wedge \ X \cap Y \subseteq Y
\end{array}
$$

For m^{-1}, however, equivalence (20) gives the reverse inclusion: yielding, in pointwise style

$$m^{-1}\,(Y \cap Z) \ = \ m^{-1}\,Y \cap m^{-1}\,Z \tag{34}$$

The pointfree proof is as follows:

$$
\begin{array}{ll}
 & m^{-1} \cdot \cap \ = \ \cap \cdot (m^{-1} \times m^{-1}) \\
\Leftrightarrow & \quad \{ \ \text{definition of } m^{-1} \ \} \\
 & \Lambda(m^{\circ} \cdot \supseteq) \cdot \cap \ = \ \cap \cdot (\Lambda(m^{\circ} \cdot \supseteq) \times \Lambda(m^{\circ} \cdot \supseteq)) \\
\Leftrightarrow & \quad \{ \ (24) \ \} \\
 & \Lambda(m^{\circ} \cdot \supseteq \cdot \cap) \ = \ \cap \cdot (\Lambda(m^{\circ} \cdot \supseteq) \times \Lambda(m^{\circ} \cdot \supseteq))
\end{array}
$$

$$
\begin{array}{ll}
\Leftrightarrow & \{\ (20)\ \} \\
& m^{\circ}\cdot \supseteq \cdot \cap\ = \ni \cdot \cap \cdot (\Lambda(m^{\circ}\cdot \supseteq) \times \Lambda(m^{\circ}\cdot \supseteq) \\
\Leftrightarrow & \{\ \text{laws: } \star\cdot \cup = \Delta^{\circ}\cdot(\star\times\star), \text{ where } \star := \ni, \supseteq, \text{ for } \Delta x = \langle x, x\rangle\ \} \\
& m^{\circ}\cdot\Delta^{\circ}\cdot(\supseteq\times\supseteq)\ =\ \Delta^{\circ}\cdot(\ni\times\ni)\cdot(\Lambda(m^{\circ}\cdot \supseteq) \times \Lambda(m^{\circ}\cdot \supseteq)) \\
\Leftrightarrow & \{\ \times \text{ is a functor and } \Delta \text{ a natural transformation } \} \\
& \Delta^{\circ}\cdot(m^{\circ}\times m^{\circ})\cdot(\supseteq\times\supseteq)\ =\ \Delta^{\circ}\cdot(\ni\cdot\Lambda(m^{\circ}\cdot \supseteq \times \ni\cdot\Lambda(m^{\circ}\cdot \supseteq)) \\
\Leftrightarrow & \{\ \times \text{ is a functor and (23) } \} \\
& \Delta^{\circ}\cdot(m^{\circ}\times m^{\circ})\cdot(\supseteq\times\supseteq)\ =\ \Delta^{\circ}\cdot((m^{\circ}\cdot \supseteq) \times (m^{\circ}\cdot \supseteq))
\end{array}
$$

The auxiliary result $\star\cdot\cup = \Delta^{\circ}\cdot(\star\times\star)$, where $\star := \ni, \supseteq$, is easy to verify. For example, the case $\star = \ni$ holds because

$$
\begin{array}{ll}
& \langle C_1, C_2\rangle\,(\Delta^{\circ}\cdot \ni \times \ni)\,x \\
= & \{\ \text{definition of the diagonal } \Delta x = \langle x, x\rangle \text{ and composition } \} \\
& \langle C_1, C_2\rangle\,(\ni\times\ni)\,\langle x, x\rangle\ \wedge\ \langle x, x\rangle\,\Delta^{\circ}\,x \\
= & \{\ \text{simplifying } \} \\
& C_1 \ni x\ \wedge\ C_2 \ni x \\
= & \{\ \text{intersection } \} \\
& C_1 \cap C_2\ \ni\ x \\
= & \{\ \text{relational composition } \} \\
& \langle C_1, C_2\rangle\,(\ni\cdot\cap)\,x
\end{array}
$$

These results extend, of course, to families of sets over a universe U:

$$m^{-1}\,(\bigcap_{i\in I} F_i)\ =\ \bigcap_{i\in I} m^{-1}\,F_i \tag{35}$$

$$m^{*}\,(\bigcap_{i\in I} F_i)\ \subseteq\ \bigcap_{i\in I} m^{*}\,F_i \tag{36}$$

The following two lemmas characterize the direct and inverse image of a Kleisli composition, respectively. They will be required later to establish a category of interpretations.

LEMMA 2.1. *Consider multifunctions* $m : A \prec\!\!\!\circ B$ *and* $n : B \prec\!\!\!\circ C$. *Then,*

$$(n \bullet m)^{*}\ =\ n^{*}\cdot m^{*} \tag{37}$$

PROOF.

$$
\begin{array}{ll}
 & (n \bullet m)^* \\
= & \quad \{ \text{ definition of } \bullet \ \} \\
 & (\bigcup \cdot \mathscr{P} n \cdot m)^* \\
= & \quad \{ \text{ definition of direct image of a multifunction } \} \\
 & \bigcup \cdot \mathscr{P} \bigcup \cdot \mathscr{P} \mathscr{P} n \cdot \mathscr{P} m \\
= & \quad \{ \ \bigcup \text{ is a natural transformation } \} \\
 & \bigcup \cdot \mathscr{P} n \cdot \bigcup \cdot \mathscr{P} m \\
= & \quad \{ \text{ definition of direct image } \} \\
 & n^* \cdot m^*
\end{array}
$$

■

LEMMA 2.2. *Consider multifunctions* $m : A \prec\!\!\!\triangleleft B$ *and* $n : B \prec\!\!\!\triangleleft C$*. Then,*

$$(n \bullet m)^{-1} \ = \ m^{-1} \cdot n^{-1} \tag{38}$$

PROOF. The first step relies on (20):

$$
\begin{array}{ll}
 & h = (n \bullet m)^{-1} \\
\Leftrightarrow & \quad \{ \text{ definition of } m^{-1} \text{ (31) and (21) } \} \\
 & h = \Lambda((n \bullet m)^\circ \cdot (\ni \ / \ \ni)) \\
\Leftrightarrow & \quad \{ \ (20) \ \} \\
 & \ni \cdot h = (n \bullet m)^\circ \cdot (\ni \ / \ \ni)
\end{array}
$$

We shall now prove, by indirect equality, that $\ni \cdot h = m^\circ \cdot (n^\circ \cdot (\ni \ / \ \ni)) / \ \ni)$:

$$
\begin{array}{ll}
 & X \subseteq \ni \cdot h \\
\Leftrightarrow & \quad \{ \text{ computed above } \} \\
 & X \subseteq (n \bullet m)^\circ \cdot (\ni \ / \ \ni) \\
\Leftrightarrow & \quad \{ \text{ shunting (8) } \} \\
 & (n \bullet m) \cdot X \subseteq (\ni \ / \ \ni) \\
\Leftrightarrow & \quad \{ \text{ universal property of division (10) } \}
\end{array}
$$

$$
\begin{array}{ll}
& \ni \cdot (n \bullet m) \cdot X \subseteq \ni \\
\Leftrightarrow & \quad \{ \text{ transpose cancelation (23) over (28) } \} \\
& \ni \cdot n \cdot \ni \ \cdot m \cdot X \subseteq \ni \\
\Leftrightarrow & \quad \{ \text{ universal property of division (10) } \} \\
& n \cdot \ni \cdot m \cdot X \subseteq (\ni \ / \ \ni) \\
\Leftrightarrow & \quad \{ \text{ shunting (8) } \} \\
& \ni \cdot m \cdot X \subseteq n^\circ \cdot (\ni \ / \ \ni) \\
\Leftrightarrow & \quad \{ \text{ universal property of division (10) } \} \\
& m \cdot X \subseteq (n^\circ \cdot (\ni \ / \ \ni)) / \ \ni \\
\Leftrightarrow & \quad \{ \text{ shunting (8) } \} \\
& X \subseteq m^\circ \cdot (n^\circ \cdot (\ni \ / \ \ni)) / \ \ni
\end{array}
$$

Finally, we go back to functions, by applying the power transpose, and conclude:

$$
\begin{array}{ll}
& \ni \cdot h \ = \ m^\circ \cdot (n^\circ \cdot (\ni \ / \ \ni)) / \ \ni \\
\Leftrightarrow & \quad \{ \ (20) \ \} \\
& h \ = \ \Lambda(m^\circ \cdot (n^\circ \cdot (\ni \ / \ \ni)) / \ \ni) \\
\Leftrightarrow & \quad \{ \ (23) \ \} \\
& h \ = \ \Lambda(m^\circ \cdot (\ni \cdot \Lambda((n^\circ \cdot (\ni \ / \ \ni)) / \ \ni))) \\
\Leftrightarrow & \quad \{ \ (12) \ \} \\
& h \ = \ \Lambda(m^\circ \cdot (\ni \ / \ \ni) \cdot \Lambda(n^\circ \cdot (\ni \ / \ \ni))) \\
\Leftrightarrow & \quad \{ \ (24) \ \} \\
& h \ = \ \Lambda(m^\circ \cdot (\ni \ / \ \ni)) \cdot \Lambda(n^\circ \cdot (\ni \ / \ \ni)) \\
\Leftrightarrow & \quad \{ \ \supseteq = (\ni \ / \ \ni) \ \} \\
& h \ = \ \Lambda(m^\circ \cdot \supseteq) \cdot \Lambda(n^\circ \cdot \supseteq) \\
\Leftrightarrow & \quad \{ \ \text{definition of } m^{-1} \ (31) \ \} \\
& h \ = \ m^{-1} \cdot n^{-1}
\end{array}
$$

■

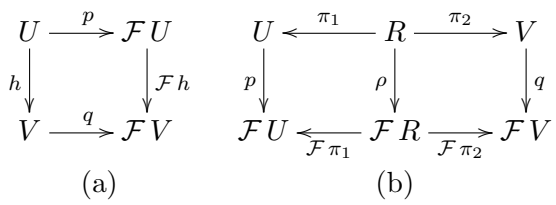


Figure 1. (a) morphism; (b) bisimulation

Coalgebras. A *coalgebra* for an endofunctor $\mathcal{F}$ in Set, often referred to as a $\mathcal{F}$-coalgebra, is a map $p : U \longrightarrow \mathcal{F}\,U$, which may be thought of as a transition structure of *shape* $\mathcal{F}$ on the set U. This shape acts as a type for the possible ways U can be observed and modified on computing p. It is also used to derive an equivalence relation (and the associated universal domain of behaviours) capturing indistinguishability by observation, which, for specific instances of $\mathcal{F}$, boils down to the notion of bisimilarity in transition systems [38] as used in computing and modal logic. Technically coalgebras are dual structures to algebras, $\mathcal{F}$ aggregating a signature of observers and, as a consequence, coinduction replacing induction as a proof principle. The increasing popularity of coalgebra theory [2, 43] in Computer Science comes exactly from its suitability to provide a generic framework to specify and reason about systems' behaviour.

A *morphism* between two $\mathcal{F}$-coalgebras $\langle U, p\rangle$ and $\langle V, q\rangle$ is a map h between the carriers U and V such that diagram (a) in Fig. 1 commutes, *i.e.*, $q \cdot h = \mathcal{F}\,h \cdot p$.

Given a subset U' of the carrier U of a coalgebra $\langle U, p\rangle$ and a map $p' : U' \longrightarrow \mathcal{F}\,U'$ such that the inclusion $i : U' \hookrightarrow U$ is a coalgebra morphism from $\langle U', p'\rangle$ to $\langle U, p\rangle$, $\langle U', p'\rangle$ is a *subcoalgebra* of $\langle U, p\rangle$.

A relation on the carriers U and V is a *bisimulation* for $\mathcal{F}$ if it can be extended to a coalgebra ρ such that projections π_1 and π_2 lift to morphisms, as expressed by the commutativity of diagram (b) in Fig. 1. An alternative definition, often more convenient in proofs, characterizes a bisimulation as a relation R such that

$$u\,R\,v \;\Rightarrow\; (p\,u)\,\widehat{\mathcal{F}R}\,(q\,v) \tag{39}$$

where $\widehat{\mathcal{F}R}$ denotes the lifting of R through $\mathcal{F}$ [20]. By eliminating variables in (39) and applying the shunting rule (8), (39) becomes

$$q \cdot R \;\subseteq\; \widehat{\mathcal{F}R} \cdot p \tag{40}$$

A basic result in coalgebra is that any morphism, regarded as a binary relation, is a bisimulation.

3. Logical interpretations and their properties

3.1. Setting the scene

Abstract logics are pairs $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$, where A is a set and $C_{\mathcal{A}}$ a closure operator on A. Such an elementary characterization is enough for our purposes. Although it will be adopted for most of what follows, all constructions and results lift smoothly to a more structured, "realistic" setting, where logics are considered over algebras with non empty signatures. It provides an elementary formalization of the concept of a *logic* as a consequence relation $\vdash_A$ captured by $C_{\mathcal{A}}$: for $\{x\} \cup X \subseteq A$, $X \vdash_A x \Leftrightarrow x \in C_{\mathcal{A}} X$.

A *closure operator* $C : \mathscr{P} A \longrightarrow \mathscr{P} A$ is defined by

$$X \subseteq C\,Y \;\;\Leftrightarrow\;\; C\,X \subseteq C\,Y \tag{41}$$

or, equivalently, by properties, (i) $X \subseteq C\,X$; (ii) $X \subseteq Y \Rightarrow C\,X \subseteq C\,Y$ and (iii) $C\,(C\,X) = C\,X$. Over the same set A, closure operators are in one-to-one correspondence with *closure systems*, *i.e.*, families of subsets of A closed under arbitrary intersections (including $\bigcap \emptyset = A$). This abstracts the well-known fact of a logic being defined by its theories. We have then another representation of $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ as $\langle A, \mathcal{T}_{\mathcal{A}} \rangle$, for $\mathcal{T}_{\mathcal{A}}$ a closure system. The relevant bijection belongs to the folklore: the closed sets of $C_{\mathcal{A}}$ give rise to $\mathcal{T}_{\mathcal{A}}$, whereas the intersection of the subset of $\mathcal{T}_{\mathcal{A}}$ containing X defines $C_{\mathcal{A}}$ on X. Formally,

$$\begin{array}{rcl} C_{\mathcal{A}} & \mapsto & \mathcal{T}_{\mathcal{A}} := \{X \subseteq A \,|\, C_{\mathcal{A}}\, X = X\} \\ \mathcal{T}_{\mathcal{A}} & \mapsto & C_{\mathcal{A}}\, X := \bigcap \{T \in \mathcal{T}_{\mathcal{A}} \,|\, X \subseteq T\}. \end{array}$$

The following, also well-known result shows how $\mathcal{A}$ is captured in a dual way by $C_{\mathcal{A}}$ or $\mathcal{T}_{\mathcal{A}}$,

Lemma 3.1. *Given two abstract logics $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{A}' = \langle A, C'_{\mathcal{A}} \rangle$ and the corresponding closure systems $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}'_{\mathcal{A}}$ over A,*

$$C_{\mathcal{A}} \,\dot{\subseteq}\, C'_{\mathcal{A}} \;\;\Leftrightarrow\;\; \mathcal{T}'_{\mathcal{A}} \subseteq \mathcal{T}_{\mathcal{A}} \tag{42}$$

Proof. From left to right, let $X \in \mathcal{T}'_{\mathcal{A}}$. By hypothesis $C_{\mathcal{A}}\, X \subseteq C'_{\mathcal{A}}\, X = X$, which, as $C_{\mathcal{A}}$ is a closure operator, yields $C_{\mathcal{A}}\, X = X$. For the opposite direction, it is enough to notice that for all $X \subseteq A$, $X \subseteq C'_{\mathcal{A}}\, X$ and, therefore, $C_{\mathcal{A}}\, X \subseteq C_{\mathcal{A}}\, (C'_{\mathcal{A}}\, X) = C'_{\mathcal{A}}\, X$. ∎

A morphism $h:\langle A, C_{\mathcal{A}}\rangle \longrightarrow \langle B, C_{\mathcal{B}}\rangle$ between abstract logics is a function from A to B preserving the consequence relation associated with the closure operator $C_{\mathcal{A}}$, *i.e.*, $x \in C_{\mathcal{A}}\, X \Rightarrow h\, x \in C_{\mathcal{B}}\,(\mathscr{P}h\, X)$. If this relation is also *reflected* back, *i.e.*,

$$x \in C_{\mathcal{A}}\, X \;\;\Leftrightarrow\;\; h\, x \in C_{\mathcal{B}}\,(\mathscr{P}h\, X) \tag{43}$$

h is called a *strict* morphism and plays a main role in relating logics. Note that (43) is equivalent to

$$C_{\mathcal{A}} \;\;=\;\; h^{-1} \cdot C_{\mathcal{B}} \cdot \mathscr{P}h \tag{44}$$

Strict morphisms admit yet another characterization in terms of closure systems.

LEMMA 3.2. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}}\rangle$ be two abstract logics. If $h:\mathcal{A} \longrightarrow \mathcal{B}$ is a strict epimorphism then*

$$\mathcal{T}_{\mathcal{A}} \;\;=\;\; \{h^{-1}\, Y \mid Y \in \mathcal{T}_{\mathcal{B}}\} \tag{45}$$

PROOF. We want to show that, for any $Y \subseteq B$, $Y \in \mathcal{T}_{\mathcal{B}}$ iff $h^{-1}\, Y \in \mathcal{T}_{\mathcal{A}}$, *i.e.*, by definition of $\mathcal{T}_{\mathcal{A}}$, $\mathcal{T}_{\mathcal{B}}$, $C_{\mathcal{A}}(h^{-1}\, Y) = h^{-1}\, Y$ iff $C_{\mathcal{B}}\, Y = Y$. Thus,

$$\begin{array}{ll} & C_{\mathcal{A}}(h^{-1}\, Y) \;=\; h^{-1}\, Y \\ \Leftrightarrow & \qquad \{\;\; (44)\;\} \\ & (h^{-1} \cdot C_{\mathcal{B}} \cdot \mathscr{P}h \cdot h^{-1})Y \;=\; h^{-1}\, Y \\ \Leftrightarrow & \qquad \{\; h \text{ is an epimorphism } \} \\ & h^{-1}(C_{\mathcal{B}}\, Y) \;=\; h^{-1}\, Y \\ \Leftrightarrow & \qquad \{\; h \text{ is an epimorphism and Leibniz rule } \} \\ & C_{\mathcal{B}}\, Y \;=\; Y \end{array}$$

Note that $h^{-1}(C_{\mathcal{B}}\, Y) \;=\; h^{-1}\, Y$ is implied by $C_{\mathcal{B}}\, Y \;=\; Y$. On the other hand it implies $(\mathscr{P}h \cdot h^{-1} \cdot C_{\mathcal{B}})Y \;=\; \mathscr{P}h(h^{-1}\, Y)$ which is equivalent to $C_{\mathcal{B}}\, Y \;=\; Y$ whenever h is an epimorphism, ■

3.2. Interpretations

As mentioned in the Introduction, the notion of logical translation we have been studying as a formal tool for program refinement [29] generalizes conservative translations to multifunctions which still preserve and reflect consequence. Formally,

DEFINITION 3.1 (Interpretation). Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$ be two abstract logics. A multifunction $m : A \prec\!\!\!\!\circ B$ is an *interpretation* from $\mathcal{A}$ to $\mathcal{B}$, if for any $\{x\} \cup X \subseteq A$,

$$x \in C_{\mathcal{A}}\, X \;\; \Leftrightarrow \;\; m\, x \subseteq C_{\mathcal{B}}\, (m^*\, X) \tag{46}$$

Clearly, by (32), definition (46) is equivalent to

$$C_{\mathcal{A}} \;\; = \;\; m^{-1} \cdot C_{\mathcal{B}} \cdot m^* \tag{47}$$

A multifunction m which preserves, but does not reflect, logical consequence will be called a *translation* in the sequel. Formally, m is a translation if

$$x \in C_{\mathcal{A}}\, X \;\; \Rightarrow \;\; m\, x \subseteq C_{\mathcal{B}}\, (m^*\, X) \tag{48}$$

or, alternatively, if $C_{\mathcal{A}} \mathrel{\dot\subseteq} m^{-1} \cdot C_{\mathcal{B}} \cdot m^*$.

Abstract logics and interpretations form a category LIntp with composition and unity arrows inherited from the Kleisli category for the powerset monad.

LEMMA 3.3. LIntp *is a category.*

PROOF. Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$, $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$ and $\mathcal{C} = \langle C, C_{\mathcal{C}} \rangle$ be abstract logics. Clearly, for any A, the identity arrow $\eta : A \prec\!\!\!\!\circ A$ in $\mathsf{Kl}(\mathscr{P})$ is an interpretation from $\mathcal{A}$ to itself. Thus, it is enough to show that equivalence (46) is preserved by Kleisli composition, *i.e.*, composing interpretations yields an interpretation. Let $m : A \prec\!\!\!\!\circ B$ and $n : B \prec\!\!\!\!\circ C$ be interpretations from $\mathcal{A}$ to $\mathcal{B}$ and from $\mathcal{B}$ to $\mathcal{C}$, respectively. Then,

$$\begin{array}{ll} & (n \bullet m)^{-1} \cdot C_{\mathcal{C}} \cdot (n \bullet m)^* \\ = & \qquad \{ \text{ lemmas 2.1 and 2.2 } \} \\ & m^{-1} \cdot n^{-1} \cdot C_{\mathcal{C}} \cdot n^* \cdot m^* \\ = & \qquad \{ \; n \text{ is an interpretation } \} \\ & m^{-1} \cdot C_{\mathcal{B}} \cdot m^* \\ = & \qquad \{ \; m \text{ is an interpretation } \} \\ & C_{\mathcal{A}} \end{array}$$

■

Some particular classes of multifunctions play an important role in the sequel. Thus,

DEFINITION 3.2. A multifunction $m : A \prec\!\!\!\leq B$, relating abstract logics $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$, is *closed* if it maps closed sets into closed sets, *i.e.*, $X = C_{\mathcal{A}}\, X \Rightarrow m^*\, X = (C_{\mathcal{B}} \cdot m^*)\, X$. It is *continuous* if $m^* \cdot C_{\mathcal{A}} \mathrel{\dot{\subseteq}} C_{\mathcal{B}} \cdot m^*$. Finally, it is *functional* whenever $m\, a$ is a singleton for all $a \in A$.

Note that a continuous multifunction is a *translation* as defined in (48), because

$$
\begin{array}{ll}
 & m^*(C_{\mathcal{A}}\, X) \;\subseteq\; C_{\mathcal{B}}(m^*\, X) \\
\Leftrightarrow & \qquad \{\ (33)\ \} \\
 & C_{\mathcal{A}}\, X \;\subseteq\; m^{-1}(C_{\mathcal{B}}(m^*\, X)) \\
\Leftrightarrow & \qquad \{\ \text{translating to predicates}\ \} \\
 & x \in C_{\mathcal{A}}\, X \;\Rightarrow\; x \in m^{-1}(C_{\mathcal{B}}(m^*\, X)) \\
\Leftrightarrow & \qquad \{\ (32)\ \} \\
 & x \in C_{\mathcal{A}}\, X \;\Rightarrow\; m\, x \subseteq C_{\mathcal{B}}(m^*\, X)
\end{array}
$$

LEMMA 3.4. *Abstract logics with closed and continuous interpretations form a subcategory of* LIntp.

PROOF. Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$, $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$ and $\mathcal{C} = \langle C, C_{\mathcal{C}} \rangle$ be abstract logics. Lemma 2.1 is all we need to prove that closedness (and continuity) is preserved through Kleisli composition. Suppose $X = C_{\mathcal{A}}\, X$. Then

$$
\begin{array}{ll}
 & X \;=\; C_{\mathcal{A}}\, X \\
\Rightarrow & \qquad \{\ m \text{ closed}\ \} \\
 & m^* X \;=\; C_{\mathcal{B}}(m^* X) \\
\Rightarrow & \qquad \{\ n \text{ closed}\ \} \\
 & (n^* \cdot m^*)X \;=\; C_{\mathcal{C}}((n^* \cdot m^*)X) \\
\Rightarrow & \qquad \{\ (37) \text{ from lemma 2.1}\ \} \\
 & (n \bullet m)X \;=\; C_{\mathcal{C}}((n \bullet m)X)
\end{array}
$$

The proof for continuity is similar. ∎

LEMMA 3.5. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$ be two abstract logics and $m : A \prec\!\!\!\leq B$ a functional and injective multifunction. Then, if m is closed and continuous wrt $\mathcal{A}$ and $\mathcal{B}$ then m is an interpretation from $\mathcal{A}$ to $\mathcal{B}$.*

PROOF. Inclusion $m^*(C_{\mathcal{A}}\,X) \subseteq C_{\mathcal{B}}(m^*\,X)$, entails $x \in C_{\mathcal{A}}\,X \Rightarrow m\,x \subseteq C_{\mathcal{B}}(m^*\,X)$ since $x \in C_{\mathcal{A}}$ implies $m\,x \subseteq m^*\,(C_{\mathcal{A}}\,X)$. Consider now a $x \in A$ such $m\,x \in C_{\mathcal{B}}(m^*\,X)$ and suppose $C_{\mathcal{B}}(m^*\,X) = m^*\,C_{\mathcal{A}}\,X$. Hence, $m\,x \subseteq m^*\,(C_{\mathcal{A}}\,X)$ and, because m is functional, there is a $y \in C_{\mathcal{A}}\,X$ such $m\,x = m\,y$. Thus, by injectivity of m, it follows that $x = y$ and, therefore, $m\,x \subseteq C_{\mathcal{B}}\,m^*\,X \Rightarrow x \in C_{\mathcal{A}}\,X$. ■

LEMMA 3.6. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}}\rangle$ be two abstract logics and $m : A \prec\!\!\!\!\backslash\; B$ a closed and continuous multifunction wrt $\mathcal{A}$ and $\mathcal{B}$. Then m is an interpretation from $\mathcal{A}$ into $\mathcal{B}$ iff for any closed set X wrt $\mathcal{A}$, $X = (m^{-1} \cdot C_{\mathcal{B}} \cdot m^*)\,X$.*

PROOF. Suppose that for any closed set X wrt $\mathcal{A}$, $X = m^{-1}\,C_{\mathcal{B}}(m^*\,X)$. Then,

$$
\begin{array}{ll}
 & C_{\mathcal{A}}\,X \\
= & \quad \{\ C_{\mathcal{A}}\,X \text{ is closed }\} \\
 & (m^{-1} \cdot C_{\mathcal{B}} \cdot m^* \cdot C_{\mathcal{A}})\,X \\
= & \quad \{\ m \text{ is closed and continuous }\} \\
 & (m^{-1} \cdot C_{\mathcal{B}} \cdot C_{\mathcal{B}} \cdot m^*)\,X \\
= & \quad \{\ C_{\mathcal{B}} \text{ is a closure operation }\} \\
 & (m^{-1} \cdot C_{\mathcal{B}} \cdot m^*)\,X
\end{array}
$$

■

We close this sub-section with a result which plays an important role in section 4 to relate interpretations and coalgebra morphisms.

LEMMA 3.7. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}}\rangle$ be two abstract logics and $m : A \prec\!\!\!\!\backslash\; B$ an interpretation. Then, $\mathcal{T}_{\mathcal{A}} = \{m^{-1}\,T \mid T \in \mathcal{T}_{\mathcal{B}}\}$.*

PROOF. Let $T \in \mathcal{T}_{\mathcal{A}}$. Since m is an interpretation, by Lemma 3.1 $T = C_{\mathcal{A}}\,T = (m^{-1} \cdot C_{\mathcal{B}} \cdot m^*)\,T$. Since $(C_{\mathcal{B}} \cdot m^*)\,T \in \mathcal{T}_{\mathcal{B}}$, $T \in \{m^{-1}\,T \mid T \in \mathcal{T}_{\mathcal{B}}\}$.

For the reverse implication, suppose $X = m^{-1}\,T$ for some $T \in \mathcal{T}_{\mathcal{B}}$. On the one hand $C_{\mathcal{A}}(m^{-1}\,T) = (m^{-1} \cdot C_{\mathcal{B}} \cdot m^* \cdot m^{-1})\,T \subseteq m^{-1}\,T$. The last inclusion holds since $(m^* \cdot m^{-1})\,T \subseteq T$, which implies $(C_{\mathcal{B}} \cdot m^* \cdot m^{-1})\,T \subseteq C_{\mathcal{B}}\,T$. Thus, $(C_{\mathcal{A}} \cdot m^{-1})\,T = m^{-1}\,T$, since $m^{-1}\,T \subseteq (C_{\mathcal{A}} \cdot m^{-1})\,T$ always. Therefore, X is closed and consequently $X \in \mathcal{T}_{\mathcal{A}}$. ■

3.3. Interpretations and congruences

This sub-section generalizes to interpretations results obtained by H. A. Feitosa in [14, 15] on conservative translations. In particular, we show that there exists an interpretation between two abstract logics iff there exists an interpretation between its associated quotient logics induced by the Frege relation. Our starting point is the following definition,

DEFINITION 3.3. Given an abstract logic $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and a multifunction $m : A \prec\!\!\!\triangleleft B$, the *abstract logic co-induced by m and $\mathcal{A}$ in B* is $\mathcal{A}_{m,B} = \langle B, C_{m,B} \rangle$, where $C_{m,B}$ is the closure operator defined for any $Y \subseteq B$ by $Y \in \mathcal{T}_{m,B}$ if $m^{-1}\, Y \in \mathcal{T}_{\mathcal{A}}$ ($\mathcal{T}_{m,B}$ denoting, as usual, the set of closed sets of $C_{m,B}$, its closure system).

Dually, given an abstract logic $\mathcal{B} = \langle B, C_{\mathcal{B}} \rangle$ and a multifunction $m : A \prec\!\!\!\triangleleft B$ the *abstract logic induced by m and $\mathcal{B}$ in A* is the abstract logic $\mathcal{B}_{A,m} = \langle A, C_{A,m} \rangle$, where $C_{A,m}$ satisfies, for any $X \subseteq A$, $X \in \mathcal{T}_{A,m}$ if $X = m^{-1}\, Y$ for some $Y \in \mathcal{T}_{\mathcal{B}}$. Whenever clear from the context we will write C_m rather than $C_{m,B}$ or $C_{A,m}$ and similarly for $\mathcal{T}_{m,B}$ or $\mathcal{T}_{A,m}$.

Note that both $\mathcal{A}_{m,B}$ and $\mathcal{B}_{A,m}$ are indeed abstract logics. To prove this it is enough to show that $\mathcal{T}_{m,B}$ and $\mathcal{T}_{A,m}$ are closed under intersections. For $\mathcal{T}_{m,B}$, let $\{Y_i\}_{i \in I}$ be a family of sets of sets in $\mathcal{T}_{m,B}$. Then $m^{-1}\, Y_i \in \mathcal{T}_{\mathcal{A}}$ for each $i \in I$. Clearly, by (35)

$$m^{-1}\,(\bigcap_{i \in I} Y_i) \in \mathcal{T}_{\mathcal{A}} \;\Leftrightarrow\; (\bigcap_{i \in I} m^{-1}\, Y_i) \in \mathcal{T}_{\mathcal{A}} \tag{49}$$

which holds because $\mathcal{T}_{\mathcal{A}}$ is a closure system. Therefore $\mathcal{T}_{m,B}$ is also a closure system. A similar proof establishes $\mathcal{T}_{A,m}$ is a closure system as well.

To discuss the relationship between quotients and interpretations, let us recall some basic notation. Let θ be an equivalence relation on a set A. As usual, θ-equivalence classes are defined as $[x]_\theta = \{y \mid x\, \theta\, y\}$, a notation which extends to sets $X \subseteq A$ as $[X]_\theta = \{[x]_\theta \mid x \in X\}$. A particular case of the latter is the quotient set $A_\theta = \{[x]_\theta \mid x \in A\}$. The map $e_\theta : A \to A_\theta$, defined by $e_\theta(a) = [a]_\theta$, is called the *canonical map*. In the sequel this will be trivially embedded in LIntp as $\eta e_\theta : A \prec\!\!\!\triangleleft A_\theta$ given by $\eta e_\theta = \eta \cdot e_\theta$, according to a convention fixed in section 2.

Consider now an abstract logic $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and an equivalence relation $\equiv$ on A. The abstract logic co-induced by $\eta e_\equiv$ will be denoted by $\mathcal{A}_\equiv$. This is characterized in the following lemma.

LEMMA 3.8. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ be an abstract logic, $\equiv$ an equivalence on A and $e_\equiv$ the corresponding canonical map. Let also $\mathcal{A}_\equiv = \langle A_\equiv, C_{\eta e_\equiv} \rangle$ be the*

logic co-induced by $\eta e_{\equiv} : A \preccurlyeq A_{\equiv}$ *and* $\mathcal{A}$ *in* $A_{\equiv}$*. Then,*

$$\mathcal{T}_{\eta e_{\equiv}} = \{[T]_{\equiv} \mid T \in \mathcal{T}_{\mathcal{A}}\}$$

PROOF. Let $B \in \mathcal{T}_{\eta e_{\equiv}}$. Then $B = (\eta e_{\equiv})^{*}\,((\eta e_{\equiv})^{-1}\,B) = [(\eta e_{\equiv})^{-1}\,B]_{\equiv}$ and $(\eta e_{\equiv})^{-1}\,B \in \mathcal{T}_{\mathcal{A}}$. Conversely, let $B \in \mathcal{T}_{\mathcal{A}}$. Then $(\eta e_{\equiv})^{-1}\,B_{\equiv} \in \mathcal{T}_{\mathcal{A}}$ and, by definition of $C_{\eta e_{\equiv}}$, $B_{\equiv} \in \mathcal{T}_{\eta e_{\equiv}}$. ■

DEFINITION 3.4 (Frege relation). Given an abstract logic $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$, the Frege relation on $\mathcal{A}$ is defined as

$$\sim_{\mathcal{A}} = \{\langle a, a'\rangle \in A \times A | C_{\mathcal{A}}\,\{a\} = C_{\mathcal{A}}\,\{a'\}\}.$$

Clearly $\sim_{\mathcal{A}}$ is an equivalence relation (whenever clear from the context the subscript will be dropped).

COROLLARY 3.5. *Given an abstract logic* $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ *and its Frege relation* $\sim$*, the logic co-induced by* $\mathcal{A}$ *and* $\eta e_{\sim}$ *in* $A_{\sim}$ *is* $\mathcal{A}_{\sim} = \langle A_{\sim}, \{[T]_{\sim} \mid T \in \mathcal{T}_{\mathcal{A}}\}\rangle$*.*

LEMMA 3.9. *Let* $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ *and* $\mathcal{B} = \langle B, C_{\mathcal{B}}\rangle$ *be two abstract logics and* m *a translation between them. Then, for any* $x, y \in A$*,*

$$C_{\mathcal{A}}\,\{x\} = C_{\mathcal{A}}\,\{y\} \;\Rightarrow\; C_{\mathcal{B}}\,m\,x = C_{\mathcal{B}}\,m\,y$$

PROOF. Suppose $C_{\mathcal{A}}\,\{x\} = C_{\mathcal{A}}\,\{y\}$. Since $C_{\mathcal{A}}$ is a closure operator, $x \in C_{\mathcal{A}}\,\{y\}$. Hence, $m\,x \subseteq C_{\mathcal{B}}\,m\,y$ and, moreover, $C_{\mathcal{B}}\,m\,y = C_{\mathcal{B}}\,(C_{\mathcal{B}}\,m\,y)$. Therefore $C_{\mathcal{B}}\,m\,x \subseteq C_{\mathcal{B}}\,m\,y$, again by definition of a closure operator. The proof of reverse inclusion is similar. ■

LEMMA 3.10. *Given an abstract logic* $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ *and an equivalence relation* $\equiv$ *on* A*,* $\eta e_{\equiv} : A \preccurlyeq A_{\equiv}$ *is a translation from* $\mathcal{A}$ *to* $\mathcal{A}_{\equiv}$*.*

PROOF. Immediate since $C_{\eta e_{\equiv}}$ is the weakest consequence that makes $\eta e_{\equiv}$ a translation. ■

We can now introduce two main results, theorems 3.6 and 3.8.

THEOREM 3.6. *For any abstract logic* $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$*,* $\eta e_{\sim} : A \preccurlyeq A_{\sim}$ *is an interpretation from* $\mathcal{A}$ *to* $\mathcal{A}_{\sim}$*.*

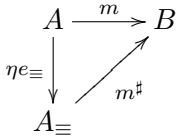
PROOF. By lemma 3.10, $\eta e_{\sim}$, for the Frege relation $\sim$, is a translation. Now suppose $[x]_{\sim} \in C_{\mathcal{A}_{\sim}}\,([X]_{\sim})$. Suppose also that for any $y \in [x]_{\sim}$, $y \notin C_{\eta e_{\sim}}\,X$, i.e., there is at least a $T \in \mathcal{T}_{\mathcal{A}}$ such $X \subseteq T$ and $y \notin T$. Then, $[y]_{\sim} \notin [T]_{\sim}$ and $[X]_{\sim} \subseteq [T]_{\sim}$. However, this is an absurd since $[x]_{\sim} = [y]_{\sim}$, and $[x]_{\sim} \in C_{\eta e_{\sim}}\,([X]_{\sim})$ which implies that $[y]_{\sim} = [x]_{\sim} \in [T]_{\sim}$. Therefore, there is $y \in [x]_{\sim}$ such that $y \in C_{\mathcal{A}}\,X$. Since $C_{\mathcal{A}}\,\{x\} = C_{\mathcal{A}}\,\{y\}$, it follows that $x \in C_{\mathcal{A}}\,X$. ■

DEFINITION 3.7. A multifunction $m : A \prec\!\!\!\!\circ A$ is *compatible* with a binary relation R if $R \subseteq m^\circ \cdot \mathscr{P}R \cdot m$, *i.e.*,

$$a\,R\,a' \;\Rightarrow\; m\,a\,(\mathscr{P}R)\,m\,a'$$

where $X\,(\mathscr{P}R)\,Y \Leftrightarrow \forall_{x\in X, y\in Y}\;\; xRy$.

LEMMA 3.11. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}}\rangle$ and $\mathcal{B} = \langle B, C_{\mathcal{B}}\rangle$ be two abstract logics and $m : A \prec\!\!\!\!\circ B$ a translation between them. If m is compatible with a relation $\equiv$, then there is a unique $m^\sharp : A_\equiv \prec\!\!\!\!\circ B$ making the following diagram in* LIntp *commute.*



i.e., $m^\sharp \bullet \eta e_\equiv = m$. *Moreover, $m^\sharp$ is a translation from $\mathcal{A}_\equiv$ to $\mathcal{B}$.*

PROOF. Since m is compatible with $\equiv$ we can define the multifunction $m^\sharp : A_\equiv \prec\!\!\!\!\circ B$ by $m^\sharp\,([x]_\equiv) = m\,x$. Therefore $(m^\sharp \bullet \eta e_\equiv)\,x = m^\sharp\,([x]_\equiv) = m\,x$. Hence the diagram commutes.

For the uniqueness of $m^\sharp$, let $g : A_\equiv \prec\!\!\!\!\circ B$ be such that $g \bullet \eta e_\equiv = m$. Since $e_\equiv$ is surjective $e_\equiv \cdot e_\equiv^{-1} = id_{A_\equiv}$. Then, by (18), $\eta e_\equiv \bullet \eta e_\equiv^{-1} = \eta id_{A_\equiv} = \eta$. Therefore, unfolding the relevant definitions and resorting to (19), yields

$$\begin{array}{ll}
& g \\
= & \quad \{\ e_\equiv \text{ is surjective and (18)}\ \} \\
& g \bullet (\eta e_\equiv \bullet \eta e_\equiv^{-1}) \\
= & \quad \{\ \bullet \text{ associative and } m = g \bullet \eta e_\equiv\ \} \\
& m \bullet \eta e_\equiv^{-1} \\
= & \quad \{\ m = m^\sharp \bullet \eta e_\equiv\ \} \\
& (m^\sharp \bullet \eta e_\equiv) \bullet \eta e_\equiv^{-1} \\
= & \quad \{\ \bullet \text{ associative, } e_\equiv \text{ is surjective and (18)}\ \} \\
& m^\sharp
\end{array}$$

Finally, to prove that $m^\sharp$ is a translation, suppose $[x]_\equiv \in C_{\mathcal{A}_\equiv}\,([X]_\equiv)$. Then, there is a $y \in [x]_\equiv$ such that $y \in C_{\mathcal{A}}\,X$ and, since m is translation, $m\,y \subseteq C_{\mathcal{B}}\,m^*\,X$. Thus,

$$m^\sharp([x]_\equiv) = m^\sharp([y]_\equiv) = (m^\sharp \bullet \eta e_\equiv)\,y = m\,y \;\subseteq\; C_{\mathcal{B}}\,m^*\,X = C_{\mathcal{B}}\,m^\sharp([x]_\equiv)$$

Therefore $m^\sharp$ is a translation. ■

THEOREM 3.8. *Let $\mathcal{A} = \langle A, C_\mathcal{A}\rangle$ and $\mathcal{B} = \langle B, C_\mathcal{B}\rangle$ be two abstract logics. Then, there exists an interpretation $m : A \preccurlyeq B$ iff there exists an interpretation $m^\sharp : A_{\sim_\mathcal{A}} \preccurlyeq B_{\sim_\mathcal{B}}$.*

PROOF. Suppose m is an interpretation. By Theorem 3.6, $\eta e_{\sim_\mathcal{B}}$ is also an interpretation and so is $\eta e_{\sim_\mathcal{B}} \bullet m$. By Lemma 3.9, $\eta e_{\sim_\mathcal{B}} \bullet m$ is compatible with $\sim_\mathcal{A}$ and, by Lemma 3.11, there exists a unique translation $m^\sharp : A_{\sim_\mathcal{A}} \preccurlyeq B_{\sim_\mathcal{A}}$ that making the following diagram in LIntp commute.

$$\begin{array}{ccc} A & \xrightarrow{\eta e_{\sim_\mathcal{A}}} & A_{\sim_\mathcal{A}} \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle m^\sharp} \\ B & \xrightarrow[\eta e_{\sim_\mathcal{B}}]{} & B_{\sim_\mathcal{B}} \end{array}$$

Since $\eta e_{\sim_\mathcal{B}} \bullet m$ is an interpretation, and $\eta e_{\sim_\mathcal{B}} \bullet m = m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$, we conclude that $m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$ is also an interpretation.

Suppose, now, that $m^\sharp$ is not an interpretation. Since, by Lemma 3.9, $m^\sharp$ is a translation, there exists $\{y\} \cup Y \subseteq A_{\sim_\mathcal{A}}$ such that $m^\sharp\, y \subseteq C_{\sim_\mathcal{B}}\,(m^{\sharp^*}\, Y)$ and $y \notin C_{\sim_\mathcal{A}}\,(Y)$ (notice that, to simplify notation, $C_{\eta e_{\sim_\mathcal{Z}}}$ is represented by $C_{\sim_\mathcal{Z}}$ for any $\mathcal{Z}$). Suppose that $m^\sharp\, y \subseteq C_{\sim_\mathcal{B}}(m^{\sharp^*}\, Y)$. Since $e_{\sim_\mathcal{A}}$ is surjective, there exists $\{x\} \cup X \subseteq A$ such that $e_{\sim_\mathcal{A}}\, x = y$ and $\mathscr{P}(e_{\sim_\mathcal{A}})\, X = Y$. Hence, $(m^\sharp \bullet \eta e_{\sim_\mathcal{A}})x \subseteq C_{\sim_\mathcal{B}}((m^{\sharp^*} \bullet \eta e^*_{\sim_\mathcal{A}})\, X) = C_{\sim_\mathcal{B}}((m^\sharp \bullet \eta e_{\sim_\mathcal{A}})^*\, X)$. Since $m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$ is an interpretation, $x \in C_{\sim_\mathcal{A}}\, X$ and so $\eta e_{\sim_\mathcal{A}}\, x \subseteq C_{\sim_\mathcal{A}}\,(\eta e_{\sim_\mathcal{A}}\, X)$, *i.e.*, $y \in C_\mathcal{A}\, Y$, which is absurd.

To prove the reverse implication, suppose that $m^\sharp$ is an interpretation. Let $m := \eta e^{-1}_{\sim_\mathcal{B}} \bullet m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$. Then $\eta e_{\sim_\mathcal{B}} \bullet m = \eta e_{\sim_B} \bullet \eta e^{-1}_{\sim_\mathcal{B}} \bullet m^\sharp \bullet \eta e_{\sim_\mathcal{A}} \Leftrightarrow \eta e_{\sim_\mathcal{B}} \bullet m = m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$. Hence, since $m^\sharp \bullet \eta e_{\sim_\mathcal{A}}$ is an interpretation, so is $\eta e_{\sim_\mathcal{B}} \bullet m$. Then,

$$\begin{array}{ll} & x \in C_\mathcal{A}(X) \\ \Leftrightarrow & \quad \{\ \eta e_{\sim_\mathcal{B}} \bullet m \text{ is an interpretation } \} \\ & (\eta e_{\sim_\mathcal{B}} \bullet m)\, x \subseteq C_{\mathcal{B}_\sim}\, (\eta e_{\sim_\mathcal{B}} \bullet m)\, X \\ \Leftrightarrow & \quad \{\ \text{Theorem 3.6 } \} \\ & m\, x \subseteq C_\mathcal{B}(m^*\, X) \end{array}$$

Therefore m is an interpretation. ■

4. Interpretations as coalgebra morphisms

4.1. The problem

As mentioned in the Introduction, the motivation for the present paper was a theorem by A. Palmigiano [37] establishing that an abstract logic $\mathcal{A} = \langle A, \mathcal{T}_\mathcal{A} \rangle$ could be seen as a coalgebra for the contravariant functor $\mathcal{T} : \mathsf{Set} \longrightarrow \mathsf{Set}$ which maps a set A to the set of closure systems over A and, contravariantly, each function $f : A \longrightarrow B$ to

$$\begin{array}{rccl} \mathcal{T}f: & \mathcal{T}B & \to & \mathcal{T}A \\ & \mathcal{S} & \mapsto & \{f^{-1}\,X \mid X \in \mathcal{S}\} \end{array}$$

Furthermore, it is proved that strict logical morphisms correspond to morphisms connecting such coalgebras. In detail, coalgebra ξ associated to $\mathcal{A}$ maps each $a \in A$ to the subset of $\mathcal{T}_\mathcal{A}$ whose elements contain a. This is well-known to form a closure system as well, *i.e.*,

$$\begin{array}{ccc} A & & a \\ \downarrow{\scriptstyle \xi} & & \downarrow \\ \mathcal{T}A & \qquad \xi(a) = \{X \in \mathcal{T}_\mathcal{A} \mid a \in X\} & \end{array}$$

Reference [37] proceeds by characterizing the class of coalgebras coming from an abstract logic and proves, as a main result, that they form a covariety. Our quest in the present paper takes, however, a different path: logical interpretations, like strict morphisms, preserve and reflect a consequence relation. Therefore, one may ask whether a similar characterization is possible for interpretations. Framing this as a research question,

> *in which setting (*i.e.*, category) and under which conditions, can interpretations be regarded as coalgebra morphisms, mimicking the similar correspondence between coalgebra morphisms for $\mathcal{T}$ in* Set *and strict morphisms between abstract logics?*

Briefly, the answer is as follows: an interpretation m, represented by its direct image m^*, corresponds to a coalgebra morphism for an endofunctor $\overline{\mathcal{T}}$, defined below, over a category Fam of sets of sets. On the other hand, morphisms between $\overline{\mathcal{T}}$-coalgebras correspond to interpretations provided they are the direct image of continuous, closed multifunctions. Therefore, for the "only-if" part, the result is weaker than the corresponding one for strict morphisms. The remaining of this section works out the details.

Definition 4.1. The category Fam of sets of sets has as objects $\mathbb{A} = \mathscr{P}A$, for each set A and, as arrows, functions between them.

The counterpart to functor $\mathcal{T}$, described above, is an endofunctor $\overline{\mathcal{T}}$: $\mathsf{Fam} \longrightarrow \mathsf{Fam}$ which maps each $\mathbb{A}$ to $\mathscr{P}\mathbb{A}$. Functor $\overline{\mathcal{T}}$ acts contravariantly on maps: to $f : \mathbb{A} \longrightarrow \mathbb{B}$ it assigns

$$\begin{array}{rccl} \overline{\mathcal{T}} f : & \overline{\mathcal{T}}\mathbb{B} & \to & \overline{\mathcal{T}}\mathbb{A} \\ & \mathcal{S} & \mapsto & \{(f \cdot \eta)^{-1}\, T \mid T \in \mathcal{S}\} \end{array}$$

Note that for $f = m^*$, $f \cdot \eta = m$ because

$$\begin{array}{ll} & m^* \cdot \eta \\ = & \quad \{\ m^* \text{ definition } \} \\ & \bigcup \cdot \mathscr{P} m \cdot \eta \\ = & \quad \{\ \eta \text{ natural (15) } \} \\ & \bigcup \cdot \eta \cdot m \\ = & \quad \{\ (17)\ \} \\ & m \end{array}$$

Now, an abstract logic $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$ is represented by a $\overline{\mathcal{T}}$-coalgebra $\langle \mathbb{A}, \xi_{\mathcal{A}} \rangle$ given by

$$\begin{array}{ccc} \mathbb{A} & & X \\ \downarrow & & \xi_{\mathcal{A}} \downarrow \\ \overline{\mathcal{T}}\mathbb{A} & & \xi_{\mathcal{A}}\, X = \{T \in \mathcal{T}_{\mathcal{A}} \mid X \subseteq T\} \end{array}$$

The following theorems establish our main results on characterizing interpretations as coalgebra morphisms.

THEOREM 4.2. *Let $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, \mathcal{T}_{\mathcal{B}} \rangle$ be two abstract logics and $\langle \mathbb{A}, \xi_{\mathcal{A}} \rangle$, $\langle \mathbb{B}, \xi_{\mathcal{B}} \rangle$ the corresponding coalgebras. Hence, if $m : A \prec\!\!\!\cdot\, B$ is an interpretation, then its direct image m^* is a coalgebra morphism,* i.e.*, it makes the following diagram to commute:*

$$\begin{array}{ccc} \mathbb{A} & \xrightarrow{\ m^*\ } & \mathbb{B} \\ {\scriptstyle \xi_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \xi_{\mathcal{B}}} \\ \overline{\mathcal{T}}\mathbb{A} & \xleftarrow{\ \overline{\mathcal{T}} m^*\ } & \overline{\mathcal{T}}\mathbb{B} \end{array}$$

PROOF.

$$
\begin{aligned}
& \overline{\mathcal{T}} m^* \cdot \xi_{\mathcal{B}} \cdot m^* \, (X) \\
= & \qquad \{ \text{ unfolding the definitions } \} \\
& \{ m^{-1} \, T \mid T \in \{ T' \in \mathcal{T}_{\mathcal{B}} \mid m^* X \subseteq T' \} \} \\
= & \qquad \{ \text{ simplifying } \} \\
& \{ m^{-1} \, T \mid T \in \mathcal{T}_{\mathcal{B}} \wedge m^* X \subseteq T \} \\
= & \qquad \{ \text{ (33): } X \subseteq m^{-1} \, T \Leftrightarrow m^* X \subseteq T \ \} \\
& \{ m^{-1} \, T \mid T \in \mathcal{T}_{\mathcal{B}} \wedge X \subseteq m^{-1} \, T \} \\
= & \qquad \{ \text{ Theorem 3.7 } \} \\
& \{ T' \mid T' \in \mathcal{T}_{\mathcal{A}} \wedge X \subseteq T' \} \\
= & \qquad \{ \text{ definition of } \xi_{\mathcal{A}}(X) \ \} \\
& \xi_{\mathcal{A}}(X)
\end{aligned}
$$

■

The converse result does not hold in general. However,

THEOREM 4.3. *Let $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, \mathcal{T}_{\mathcal{B}} \rangle$ be two abstract logics and $m : A \prec\!\!\!\!\cdot\!\!\!\!\circ \, B$ a closed and continuous multifunction. Then, m is an interpretation if $\mathcal{T}_{\mathcal{A}} = \{ m^{-1} \, T \mid T \in \mathcal{T}_{\mathcal{B}} \}$.*

PROOF. Let X be a theory of $\mathcal{A}$, *i.e.*, a closed set of $C_{\mathcal{A}}$. Then, by hypothesis, $X = m^{-1} \, T$, for some $T \in \mathcal{T}_{\mathcal{B}}$. Hence, $m^* \, X \subseteq T$. Thus, $(C_{\mathcal{B}} \cdot m^*) \, X \subseteq C_{\mathcal{B}} \, T = T$. Therefore, $(m^{-1} \cdot C_{\mathcal{B}} \cdot m^*) \, X \subseteq m^{-1} \, T = X$. The other inclusion holds since m is continuous. ■

Therefore, a closed and continuous multifunction m such that m^* is a coalgebra morphism is an interpretation.

4.2. Reasoning in the coalgebra

What can be achieved through this coalgebraic "rephrasing" of interpretations? As it is always the case in Mathematics, whenever a different setting is proved to capture a concept, the potential gain is on new reasoning tools. The coalgebraic perspective offers bisimulation. Given two abstract logics $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$ and $\mathcal{B} = \langle B, \mathcal{T}_{\mathcal{B}} \rangle$ and a (closed, continuous) multifunction m

between them, the coalgebraic representation provides an alternative way to prove that m is an interpretation. In this case two methods are available: either one proves the direct image of m is a morphism from $\langle \mathbb{A}, \xi_{\mathcal{A}} \rangle$ to $\langle \mathbb{B}, \xi_{\mathcal{B}} \rangle$, or that its graph is a bisimulation. Using definition (39), it is easy to give an explicit representation of how a bisimulation looks like for functor $\overline{\mathcal{T}}$. Because $\overline{\mathcal{T}}$ is contravariant, a relation $R : A \longrightarrow B$ is a bisimulation iff $\xi_{\mathcal{B}} \cdot R \subseteq \widehat{\overline{\mathcal{T}}R}^{\circ} \cdot \xi_{\mathcal{A}}$, *i.e.*,

$$(X, Y) \in R \;\Rightarrow\; (\xi_{\mathcal{B}}\, Y, \xi_{\mathcal{A}}\, X) \in \widehat{\overline{\mathcal{T}}R}$$

where

$$\widehat{\overline{\mathcal{T}}R} \;=\; \{\langle \{T \in \mathcal{T}_{\mathcal{B}} \mid Y \subseteq T\}, \{T \in \mathcal{T}_{\mathcal{A}} \mid X \subseteq T\}\rangle \mid \langle X, Y\rangle \in R\}$$

In several cases, bisimulations can be tested (semi-)automatically (for example in Circ [23]).

Another useful tool in coalgebraic analysis is the notion of an *invariant*. Invariants can be characterized as coreflexive bisimulations [3]. Coreflexives (*i.e.*, relations $R : A \longrightarrow A$ such that $R \subseteq id$) are typical representations of predicates over the coalgebra carrier: to each such predicate φ over a set S corresponds a coreflexive Φ such that $\langle s, s'\rangle \in \Phi \Leftrightarrow \varphi\, s \wedge s' = s$. By (39), an invariant over a $\overline{\mathcal{T}}$-coalgebra $\xi_{\mathcal{A}}$ satisfies

$$\langle X, X\rangle \in \Phi \;\Rightarrow\; (\xi_{\mathcal{A}}\, X, \xi_{\mathcal{A}}\, X) \in \widehat{\overline{\mathcal{T}}\Phi} \tag{50}$$

i.e., φ is kept along the coalgebra structure, or equivalently, "inside the logic" modeled by $\xi_{\mathcal{A}}$. Clearly, (50) boils down to

$$\varphi\, X \;\Rightarrow\; \forall_{T \in \mathcal{T}_{\mathcal{A}}}\;\; X \subseteq T \Rightarrow \varphi\, T$$

Properties of interpretations can also be studied in terms of the corresponding coalgebra morphisms. For example, m^* being an epi coalgebra morphism is a sufficient condition for m itself to be surjective. Such reflection of epimorphisms is due to the fact that the forgetful functor from the category of $\overline{\mathcal{T}}$-coalgebras to Fam creates limits. Surjective interpretations verify

$$m^* \cdot C_{\mathcal{A}} \;=\; C_{\mathcal{B}} \cdot m^*$$

because

$$\begin{array}{ll} & C_{\mathcal{A}} \;=\; m^{-1} \cdot C_{\mathcal{B}} \cdot m^{*} \\ \Rightarrow & \qquad \{ \text{ Leibniz } \} \\ & m^{*} \cdot C_{\mathcal{A}} \;=\; m^{*} \cdot m^{-1} \cdot C_{\mathcal{B}} \cdot m^{*} \\ \Leftrightarrow & \qquad \{ \; m \text{ surjective } (m^{*} \cdot m^{-1} = id) \; \} \\ & m^{*} \cdot C_{\mathcal{A}} \;=\; C_{\mathcal{B}} \cdot m^{*} \end{array}$$

which entails a strong form of equivalence between the underlying abstract logics: for $m : A \prec\!\!\!\cdot\, B$, and $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$, $\mathcal{B} = \langle B, \mathcal{T}_{\mathcal{B}} \rangle$ not only consequence in $\mathcal{A}$ is preserved and reflected along m, as witnessed by the definition of interpretation itself (46), but, additionally, the same happens to consequence in $\mathcal{B}$ along m^{-1}:

$$Z \subseteq C_{\mathcal{B}}\, Y \;\;\Leftrightarrow\;\; m^{-1}\, Z \subseteq C_{\mathcal{A}}\, (m^{-1}\, Y) \tag{51}$$

When m boils down to a function, and therefore to a strict morphism between $\mathcal{A}$ and $\mathcal{B}$, this is called in [16] a *bilogical* morphism. The same qualifier may be used for these interpretations as well.

Some properties of the logic have direct counterparts in the coalgebra. For example, a *fragment* of an abstract logic $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ is defined as another logic $\mathcal{A} = \langle A', C_{\mathcal{A}'} \rangle$ such that $A' \subseteq A$ and for all $X \cup \{a\} \subseteq A'$, $a \in C_{\mathcal{A}}\, X \Leftrightarrow a \in C_{\mathcal{A}'}\, X$. Coalgebraically, this corresponds to a subcoalgebra construction, as follows:

LEMMA 4.1. *If $\mathcal{A}' = \langle A', C_{\mathcal{A}'} \rangle$ is a fragment of $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$, then $\xi_{\mathcal{A}'}$ is a $\mathcal{T}$-subcoalgebra of $\xi_{\mathcal{A}}$.*

PROOF. The closed sets of $\mathcal{A}'$ are the intersections of A' with the closed sets of $\mathcal{A}$. Then, if i is the inclusion $A' \subseteq A$, we have to show that $\mathcal{T}i \cdot \xi_{\mathcal{A}} \cdot i = \xi_{\mathcal{A}'}$.

$$\begin{array}{ll} & (\mathcal{T}i \cdot \xi_{\mathcal{A}} \cdot i)\, a \\ = & \qquad \{ \; \mathcal{T} \text{ and } \xi_{\mathcal{A}} \text{ definitions } \} \\ & \{ i^{-1}\, X \mid X \in \{ Y \in \mathcal{T}_{\mathcal{A}} \mid a \in Y \wedge a \in A' \} \} \\ = & \qquad \{ \text{ simplifying } \} \\ & \{ \{ a' \in A' \mid i\, a' \in Y \} \mid Y \in \mathcal{T}_{\mathcal{A}} \wedge a \in Y \wedge a \in A' \} \\ = & \qquad \{ \; i \text{ embedding and definition of } \cap \; \} \\ & \{ Y \cap A' \mid a \in Y \wedge Y \in \mathcal{T}_{\mathcal{A}} \} \end{array}$$

$$
\begin{array}{ll}
= & \{ \text{ definition of } \mathcal{T}_{\mathcal{A}'} \ \} \\
& \{Z \in \mathcal{T}_{\mathcal{A}'} \mid a \in Z\}
\end{array}
$$

■

Notice that, as i is a function, it is easier to reason directly in $\mathcal{T}$ than using, equivalently, $\eta \cdot i$ in $\overline{\mathcal{T}}$. In any case, coalgebra $\xi_{\mathcal{A}'}$, being induced by an inclusion which lifts to a coalgebra morphism, is unique, since functor $\mathcal{T}$ preserves monomorphisms. And so is, as expected, the corresponding fragment.

These examples briefly illustrate how reasoning in the coalgebraic side and transporting results back to the logical one may be a useful tool in studying consequence operators. In the next section this is further explored in the context of a specific domain of application, that of program refinement. In particular it is shown how some specific refinement situations require weaker notions of morphism between coalgebras, developed in [33], to be brought into the picture.

5. Application to program refinement

5.1. Refinement by interpretation

The design of complex software systems at ever-increasing levels of reliability is a main concern in Software Engineering. Since the 1980's research on algebraic specification methods [47, 32, 49], which resort to concepts and tools of logic (to build specifications) and universal algebra (for their models), has evolved to address such a challenge. Central to such methods, namely to CASL [36], their landmark realization, is the process of *stepwise refinement* [44, 34, 45, 26] of specifications through which a complex design is produced by incrementally adding details and reducing under-specification with respect to an initial, high-level one. This is done step-by-step until the specification becomes a precise description of a concrete model.

In this setting a specification consists of a pair $SP = \langle \Sigma_{SP}, [\![SP]\!] \rangle$ where Σ_{SP} is a signature and $[\![SP]\!]$ a class of Σ_{SP}-algebras considered to be admissible realizations of the envisaged system. The following definition recalls the basic concepts.

DEFINITION 5.1. A *signature* Σ is a pair (S, Ω), where S is a set (of sort names) and Ω is a $(S^* \times S)$-sorted set (of operation names). For a signature Σ a Σ-*algebra* A consists of an S-sorted set $A = (A_s)_{s \in S}$, where for all $s \in S$

$A_s \neq \emptyset$ and, for any $f \in \Omega_{s_1 \ldots s_n, s}$, a function $f^A : A_{s_1} \times \cdots \times A_{s_n} \to A_s$. We denote by $Alg(\Sigma)$ the class of all Σ-algebras.
Finally, a *signature morphism* from $\Sigma = (S, \Omega)$ to $\Sigma' = (S', \Omega')$ is a pair $\sigma = (\sigma_{sort}, \sigma_{op})$, 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, that is, $\sigma_{op} = (\sigma_{\omega,s} : \Omega_{\omega,s} \to \Omega'_{\sigma^*_{sort}(\omega), \sigma_{sort}(s)})_{\omega \in S^*, s \in S}$ (where for $\omega = s_1 \ldots s_n \in S^*, \sigma^*_{sort}(\omega) = \sigma_{sort}(s_1) \ldots \sigma_{sort}(s_n)$).

Given a set Φ_{SP} of requirements, a specification can be represented by $\langle \Sigma_{SP}, [\![SP]\!] \rangle$ where $[\![SP]\!] = \{A \in Alg(\Sigma_{SP}) | A \models \Phi_{SP}\}$. Implementations are derived through stepwise refinement leading to a chain

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

where, for all $1 \leq i \leq n$, $SP_{i-1} \rightsquigarrow SP_i$ means $[\![SP_i]\!] \subseteq [\![SP_{i-1}]\!]$. Each step of this process is called an *elementary refinement* step. Note that the elementary refinement relation $\rightsquigarrow$ is transitive since $\Sigma_{SP} = \Sigma_{SP'} = \Sigma_{SP''}$ and $[\![SP'']\!] \subseteq [\![SP']\!] \subseteq [\![SP]\!]$.

In practice refinement steps $SP' \rightsquigarrow SP$ are taken up to *signature morphisms*, in order to deal with renaming, adding or grouping together different signature components. Therefore, the elementary refinement step is annotated with the relevant signature morphism, say σ, and $SP' \overset{\sigma}{\rightsquigarrow} SP$ is interpreted as $[\![SP']\!] \restriction_\sigma \subseteq [\![SP]\!]$, where $[\![SP']\!] \restriction_\sigma = \{A \restriction_\sigma | A \in [\![SP]\!]\}$ and $A \restriction_\sigma$ denotes a *reduct* of the algebra A along σ.

Not all transformations relevant to software design, reuse, and adaptation are captured by signature morphisms. As mentioned in the Introduction, a more flexible refinement theory can be developed by resorting to logical interpretations instead as witnesses of refinement steps. Formally, we say that

Definition 5.2. Let SP and SP' be specifications, represented by abstract logics. We say that SP' is a *refinement by interpretation* of SP, in symbols $SP \rightharpoonup SP'$, if there is an interpretation τ followed by an elementary refinement σ connecting SP to SP', i.e,

$$SP \overset{\tau}{\prec\!\!\cdot} SP'' \overset{\sigma}{\rightsquigarrow} SP' \tag{52}$$

Characterizing logic interpretations as coalgebra morphisms, as discussed in the previous sections, provides the working software engineering with new tools to reason about refinement. Actually, such was the main motivation for this research: interpretations, captured by coalgebra morphisms, can be established by coalgebraic means.

In this section the methodology of *refinement by interpretation*, developed in [29, 30], is revisited in the coalgebraic setting. As before, we regard specifications as abstract logics $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ (or their coalgebraic counterparts $\langle \mathbb{A}, \xi_{\mathcal{A}} \rangle$), which is enough to convey the basic ideas. The introduction of further algebraic structure, although not particularly demanding, is subject of current research.

Although, as discussed earlier in the paper, logic interpretations both preserve and reflect the consequence relation, the basic requirement placed on a refinement relation, besides being a pre-order to allow stepwise construction, is preservation of properties. Taking specifications as abstract logics $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{A}' = \langle A', C_{\mathcal{A}'} \rangle$, an *elementary refinement* of $\mathcal{A}$ by $\mathcal{A}'$, denoted by $\mathcal{A} \rightsquigarrow \mathcal{A}'$, corresponds to the requirement that

$$a \in C_{\mathcal{A}}\, X \;\Rightarrow\; a \in C_{\mathcal{A}'}\, X \tag{53}$$

This can be expressed simply as $C_{\mathcal{A}} \mathrel{\dot\subseteq} C_{\mathcal{A}'}$ or, in terms of the corresponding closure systems, as $\mathcal{T}_{\mathcal{A}'} \subseteq \mathcal{T}_{\mathcal{A}}$.

Expressing $\mathcal{A} \rightsquigarrow \mathcal{A}'$ as the subcoalgebra morphism induced by the relevant inclusion $i : A \hookrightarrow A'$, *i.e.*, by the commutativity of the diagram below (for $\mathcal{T}$ defined in section 4.1), is too strong: it makes i a strict morphism, replacing implication by equivalence in (53).

$$\begin{array}{ccc} A & \xhookrightarrow{\;i\;} & A' \\ {\scriptstyle \xi_{\mathcal{A}}}\downarrow & & \downarrow{\scriptstyle \xi_{\mathcal{A}'}} \\ \mathcal{T}A & \xleftarrow{\;\mathcal{T}i\;} & \mathcal{T}A' \end{array}$$

Actually, it enforces $(\mathcal{T}i \cdot \xi_{\mathcal{A}'} \cdot i)\, a = \{i^{-1}\, X \mid X \in \xi_{\mathcal{A}'}\, a\} = \{i^{-1}\, X \mid X \in \mathcal{T}_{\mathcal{A}'} \wedge a \in X\}$ to coincide with $\xi_{\mathcal{A}}\, a = \{X \in \mathcal{T}_{\mathcal{A}} \mid a \in X\}$, and, therefore, $\mathcal{T}_{\mathcal{A}} = \mathcal{T}_{\mathcal{A}'}$.

Preservation of consequence, however, only requires $\mathcal{T}_{\mathcal{A}'} \subseteq \mathcal{T}_{\mathcal{A}}$. Coalgebraically, this means $\xi_{\mathcal{A}}$ and $\xi_{\mathcal{A}'}$ should be related simply by a *simulation*. This is captured by the following weaker notion of coalgebra morphism:

Definition 5.3. Let $\mathcal{F}$ be a contravariant functor on Set, $\langle A, \alpha \rangle$ and $\langle B, \beta \rangle$ be two $\mathcal{F}$-coalgebras and $\sqsubseteq$ a pre-order on B preserved by $\mathcal{F}$. A *forward morphism* between $\langle A, \alpha \rangle$ and $\langle B, \beta \rangle$ with respect to $\sqsubseteq$ is a map $h : A \longrightarrow B$ such that $\mathcal{F}h \cdot \beta \cdot h \mathrel{\dot\sqsubseteq} \alpha$, as depicted in the diagram below.

$$\begin{array}{ccc} A & \xrightarrow{\;h\;} & B \\ {\scriptstyle \alpha}\downarrow & \sqsupseteq & \downarrow{\scriptstyle \beta} \\ \mathcal{F}A & \xleftarrow[\;\mathcal{F}h\;]{} & \mathcal{F}B \end{array}$$

Reference [33] introduced forward morphisms of coalgebras for regular functors and proved them to preserve the underlying transition relation. For our purposes here it is enough to note that

LEMMA 5.1. *Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{A}' = \langle A', C_{\mathcal{A}'} \rangle$ two abstract logics and $\langle A, \xi_{\mathcal{A}} \rangle$ and $\langle A, \xi_{\mathcal{A}'} \rangle$ their corresponding $\mathcal{T}$-coalgebras. $\mathcal{A}'$ is an elementary refinement of $\mathcal{A}$ iff there is an inclusion $i : A \hookrightarrow A'$ which is a $\mathcal{T}$-coalgebra forward morphism wrt $\subseteq$ between $\mathcal{A}$ and $\mathcal{A}'$.*

PROOF. Immediate, by unfolding the definitions. ∎

Moreover, forward morphisms compose giving rise to a category of abstract logics and elementary refinements:

THEOREM 5.4. *The class of $\mathcal{T}$-coalgebras induced by abstract logics and $\mathcal{T}$-forward morphisms wrt $\subseteq$ form a category.*

PROOF. It is enough to show that the composition of $\mathcal{T}$-forward morphisms wrt $\subseteq$ is still a $\mathcal{T}$-forward morphism, as identities and composition are inherited from the category of $\mathcal{T}$-coalgebras. Let $\mathcal{A} = \langle A, \mathcal{T}_{\mathcal{A}} \rangle$, $\mathcal{B} = \langle B, \mathcal{T}_{\mathcal{B}} \rangle$ and $\mathcal{C} = \langle C, \mathcal{T}_{\mathcal{C}} \rangle$ be abstract logics and $\langle A, \xi_{\mathcal{A}} \rangle$, $\langle B, \xi_{\mathcal{B}} \rangle$ and $\langle C, \xi_{\mathcal{C}} \rangle$ the corresponding $\mathcal{T}$-coalgebras. Consider the situation depicted in the following diagram, where h and g are forward morphisms.

$$\begin{array}{ccccc} A & \xrightarrow{\ h\ } & B & \xrightarrow{\ g\ } & C \\ {\scriptstyle \xi_{\mathcal{A}}}\downarrow & \supseteq & \downarrow{\scriptstyle \xi_{\mathcal{B}}} & \supseteq & \downarrow{\scriptstyle \xi_{\mathcal{C}}} \\ \mathcal{T}A & \xleftarrow{\ \mathcal{T}h\ } & \mathcal{T}B & \xleftarrow{\ \mathcal{T}g\ } & \mathcal{T}C \end{array}$$

Thus,

$$\begin{array}{ll} & \mathcal{T}(h \cdot g) \cdot \xi_{\mathcal{C}} \cdot (g \cdot h) \\ = & \quad \{\ \mathcal{T} \text{ is a functor and associativity of } \cdot\ \} \\ & \mathcal{T}h \cdot ((\mathcal{T}g \cdot \xi_{\mathcal{C}} \cdot g) \cdot h) \\ \dot{\subseteq} & \quad \{\ g \text{ is a forward morphism and } s \,\dot{\subseteq}\, r \Rightarrow (t \cdot s) \,\dot{\subseteq}\, (t \cdot r)\ \} \\ & \mathcal{T}h \cdot (\xi_{\mathcal{B}} \cdot h) \\ \dot{\subseteq} & \quad \{\ h \text{ is a forward morphism and } s \,\dot{\subseteq}\, r \Rightarrow (t \cdot s) \,\dot{\subseteq}\, (t \cdot r)\ \} \\ & \xi_{\mathcal{A}} \end{array}$$

∎

Let us now turn to the general case where refinement is witnessed by an interpretation. To frame the refinement situation in definition 5.2 as a coalgebra morphism, one needs first to represent elementary refinements in Fam. Let $\mathcal{A} = \langle A, C_{\mathcal{A}} \rangle$ and $\mathcal{A}' = \langle A', C_{\mathcal{A}'} \rangle$ be two abstract logics and $\langle A, \xi_{\mathcal{A}} \rangle$ and $\langle A, \xi_{\mathcal{A}'} \rangle$ their corresponding $\mathcal{T}$-coalgebras. This is achieved by embedding the diagram

$$\begin{array}{ccc} A & \xhookrightarrow{i} & A' \\ {\scriptstyle \xi_{\mathcal{A}}}\downarrow & \supseteq & \downarrow{\scriptstyle \xi_{\mathcal{A}'}} \\ \mathcal{T}A & \xleftarrow[\mathcal{T}i]{} & \mathcal{T}A' \end{array}$$

in Fam, yielding

$$\begin{array}{ccc} \mathbb{A} & \xhookrightarrow{i^*} & \mathbb{A}' \\ {\scriptstyle \overline{\xi_{\mathcal{A}}}}\downarrow & \supseteq & \downarrow{\scriptstyle \overline{\xi_{\mathcal{A}'}}} \\ \overline{\mathcal{T}}\mathbb{A} & \xleftarrow[\overline{\mathcal{T}}i^*]{} & \overline{\mathcal{T}}\mathbb{A}' \end{array}$$

where $\mathbb{A} = \mathscr{P}A$ and defining $\overline{\xi_{\mathcal{A}}}$ such that $\eta \cdot \xi_{\mathcal{A}} = \overline{\xi_{\mathcal{A}}} \cdot \eta$. Hence, (52) translates to the commutativity of the following diagram in Fam:

$$\begin{array}{ccccc} \mathbb{A} & \xrightarrow{\tau^*} & \mathbb{C} & \xhookrightarrow{\sigma^*} & \mathbb{B} \\ {\scriptstyle \overline{\xi_{\mathcal{A}}}}\downarrow & & {\scriptstyle \overline{\xi_{\mathcal{C}}}}\downarrow & \supseteq & \downarrow{\scriptstyle \overline{\xi_{\mathcal{B}}}} \\ \overline{\mathcal{T}}\mathbb{A} & \xleftarrow[\overline{\mathcal{T}}\tau^*]{} & \overline{\mathcal{T}}\mathbb{C} & \xleftarrow[\overline{\mathcal{T}}\sigma^*]{} & \overline{\mathcal{T}}\mathbb{B} \end{array} \tag{54}$$

This establishes $\sigma^* \cdot \tau^*$ as a forward morphism in Fam. The left square commutes strictly whereas commutativity of the right one is up to set inclusion $\subseteq$. Note that refinement steps represented in diagram (54) can be composed along a refinement chain capturing the whole implementation process of a specification SP, as depicted below.

$$\begin{array}{ccccccccccccc} \mathbb{A} & \xrightarrow{\tau_0^*} & \mathbb{C}_0 & \xhookrightarrow{\sigma_0^*} & \mathbb{C}_1 & \xrightarrow{\tau_1^*} & \cdots & \xhookrightarrow{\sigma_{n-1}^*} & \mathbb{C}_n & \xrightarrow{\tau_n^*} & \mathbb{C}_n & \xhookrightarrow{\sigma_n^*} & \mathbb{B} \\ {\scriptstyle \overline{\xi_{\mathcal{A}}}}\downarrow & & {\scriptstyle \overline{\xi_{\mathcal{C}_0}}}\downarrow & \supseteq & \downarrow{\scriptstyle \overline{\xi_{\mathcal{C}_1}}} & & & & {\scriptstyle \overline{\xi_{\mathcal{C}_{n-1}}}}\downarrow & & {\scriptstyle \overline{\xi_{\mathcal{C}_n}}}\downarrow & \supseteq & \downarrow{\scriptstyle \overline{\xi_{\mathcal{B}}}} \\ \overline{\mathcal{T}}\mathbb{A} & \xleftarrow[\overline{\mathcal{T}}\tau_0^*]{} & \overline{\mathcal{T}}\mathbb{C}_0 & \xleftarrow[\overline{\mathcal{T}}\sigma_0^*]{} & \overline{\mathcal{T}}\mathbb{C}_0 & \xleftarrow[\overline{\mathcal{T}}\tau_1^*]{} & \cdots & \xleftarrow[\overline{\mathcal{T}}\sigma_{n-1}^*]{} & \overline{\mathcal{T}}\mathbb{C}_{n-1} & \xleftarrow[\overline{\mathcal{T}}\tau_n^*]{} & \overline{\mathcal{T}}\mathbb{C}_n & \xleftarrow[\overline{\mathcal{T}}\sigma_n^*]{} & \overline{\mathcal{T}}\mathbb{B} \end{array}$$

5.2. Examples

This section illustrates through a few examples the notion of *refinement by interpretation* which motivates the work reported in this paper. As mentioned above, this constitutes a new application of Algebraic Logic techniques to a Computer Science problem, which opens new perspectives to its understanding. The following examples emphasize such a potential, capturing refinement situations which are difficult to express, or simply not expressible, through signature morphisms.

To discuss examples of concrete interpretations requires specializing the abstract framework introduced in the previous sections to capture logical systems over many sorted signatures. Moreover, often software specification entails the need for different kinds of logical systems, even of different dimensions. These concerns were addressed in our previous work [29, 30] through the notion of a k-formula for a nonzero natural number k. A *k-formula* of sort S over Σ is just a sequence of k Σ-formulas, all of the same sort S.

Hidden k-logics (see [31] for their systematic study), and even, sometimes, just k-logics (see [30]) provide an interesting setting for specifications. Hidden k-logics are a natural generalization of k-deductive systems that encompass equational and inequational logics. They are defined in the usual Tarski way as a consequence relation satisfying reflexivity, cut, weakening and structural conditions. Hidden k-logics generalize k-deductive systems in two directions. On the one hand, sorts are taken into account in order to specify programs which involve different data types. The second direction, on the other hand, is computationally motivated by the need to hide in an internal 'memory' all pertinent information about the abstract machine underlying the program's execution. This is often required when specifying object-oriented programs [1]. It places, however, a special challenge to the equational methods typically used in specifications of abstract data types. This can be addressed by augmenting the standard equality predicate by *behavioral equivalence*, which, in this Abstract Algebraic Logic approach, can be achieved by means of properties of the well known *Leibniz congruence* [31].

An important class of such logics admits a presentation by axioms and inference rules in the Hilbert style. It is well known how axioms and inference rules induce a hidden logic, hence in the following examples we will just give the associated sets of axioms and inference rules of the logic.

As in the abstract case considered in the previous sections, where an interpretation is a multifunction to relate specifications axiomatized in different logical systems (*i.e.*, different k-logics) one has often to resort to mul-

tifunctions between the respective sets of formulas. This leads us to multi-functions, called (k-l)-translations in our previous work, that map k-formulas to sets of l-formulas.

Our first two examples deal with equational specifications. The third shows how the approach can be generalized to deductive systems of arbitrary dimension.

EXAMPLE 5.1. *Consider the following two specifications:*

spec *SP1*	***spec*** *SP2*
sorts	***sorts***
s	s
ops	***ops***
$f : s \to s$	$ok :\to s$
axioms	$f : s \to s$
$x \approx x$	$test : s \times s \to s$
	axioms
	$test(x, x) \approx ok$
inference rules	***inference rules***
$\dfrac{x \approx x'}{x' \approx x}$	$\dfrac{test(x, x') \approx ok}{test(x', x) \approx ok}$
$\dfrac{x \approx x', x' \approx x''}{x \approx x''}$	$\dfrac{test(x, x') \approx ok, test(x', x'') \approx ok}{test(x, x'') \approx ok}$
$\dfrac{x \approx x'}{f(x) \approx f(x')}$	$\dfrac{test(x, x') \approx ok}{test(f(x), f(x')) \approx ok}$

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

$$\begin{array}{rccc} \tau : & \mathrm{Eq}(SP1) & \to & \mathrm{Eq}(SP2) \\ & x \approx x' & \mapsto & test(x, x') \approx ok \end{array}$$

interprets SP1. Actually, since the axiomatization of SP2 is defined by the translation of SP1, we have

$$\vdash_{SP1} x \approx x' \text{ iff } \vdash_{SP2} test(x, x') \approx ok.$$

On the other hand, an inspection of the signatures of both specifications shows that there exists an unique signature morphism definable between them: the inclusion $\iota : Sig(SP1) \to Sig(SP2)$. This morphism induces the

identity translation between formulas which, obviously, does not interpret *SP1* in *SP2* .

If this example introduces a very simple refinement that is not, however, captured by translations induced by signature morphisms, the following one goes a step further. It illustrates how useful, even if not elementary, design transformations in algebraic specifications can be captured as refinements by interpretation. The example, borrowed from a Computer Science context, focus on one of such transformations in which some operations are *decomposed* or *mapped to transactions*, *i.e.*, sequences of operations to be executed atomically.

EXAMPLE 5.2. *Consider the following fragment of a specification of a bank account management system (*BAMS*), involving account deposits (operation* deposit*), withdrawals (*withdraw*) and a balance query (*bal*). Assume 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* $\mathsf{bal} : Sys \times Ac \longrightarrow Int$*.*

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

Consider, now, an implementation BAMS_{Val}*, where all debit and credit transactions require a previous validation step. This is achieved through an operation* $\mathsf{val} : Sys \times Ac \times Int \longrightarrow Int$*, 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 will return the original amount, in the second* 0 *as an error value. This will force an invalid deposit or withdrawal to have no effect (*0 *will be added or subtracted to the account's balance). The axioms for* BAMS_{VAL} *include,*

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

The interpretation $\tau_1 : \mathrm{Eq}(\Sigma_{\mathrm{BAMS}}) \prec \mathrm{Eq}(\Sigma_{\mathrm{BAMS_{VAL}}})$*, defined by*

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

$$\begin{array}{lll}
\tau_1^{\#}(x) = \{x\} & & \text{for } x \in \mathrm{VAR} \\
\tau_1^{\#}(f(t_1,t_2,t_3)) = \{f(t'_1,t'_2,\mathsf{val}(t'_1,t'_2,t'_3)) \mid \bigwedge_{i=1..3} t'_i \in \tau_1^{\#}(t_i)\} & & \text{for } f \in \{\mathsf{deposit},\mathsf{withdraw}\} \\
\tau_1^{\#}(f(t_1,\ldots,t_n)) = \{f(t'_1,\ldots,t'_n) \mid \bigwedge_{i=1..n} t'_i \in \tau_1^{\#}(t_i)\} & & \text{for } f \notin \{\mathsf{deposit},\mathsf{withdraw}\}
\end{array}$$

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

We close this section with an example capturing a *change of logic* in a more general sense.

EXAMPLE 5.3. *A semilattice can be regarded either as an algebra or as a partially order structure. Such a duality, often useful in specifications, can be expressed by an interpretation, actually an equivalence between two 2-logics over the one-sorted signature* $\Sigma = \{\wedge\}$ *(see [7]). Consider the following logics, where* EQ_Σ *stands for the (free) equational logic over* Σ*,*

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

and SLP*, the specifiable 2-logic defined by the following axioms and inference rules:*

spec 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
$$\frac{\langle x, y\rangle, \langle y, z\rangle}{\langle x, z\rangle}$$
$$\frac{\langle x_0, y_0\rangle, \langle x_1, y_1\rangle}{(x_0 \wedge x_1, y_0 \wedge y_1\rangle}$$

The schematic translation defined by the multifunction

$$\tau(\langle p, q\rangle) = \{\langle p, q\rangle, \langle q, p\rangle\}$$

witnesses that SLP *interprets* SLV*.*

6. Conclusions

Originally defined as a tool for studying equivalent algebraic semantics (see *e.g.* [6, 7, 8, 11]), the notion of *logical interpretation* proved effective in expressing a number of transformations in program refinement, difficult to deal with in classical terms. Some of these cases were illustrated by the examples discussed in section 5. The theory of *refinement by interpretation* was further developed in our previous work: first introduced for the popular equational case [29] and later generalized to deductive systems of arbitrary dimension in [30]. The latter makes possible, for example, to refine sentential into equational specifications and the latter into modal ones.

Besides illustrating the role of interpretations in program refinement, this paper characterized a formal correspondence between these and morphisms for a particular kind of coalgebras, generalizing [37]. This paves the way to the use of coalgebraic results and methods (namely bisimulations) in reasoning about program refinement.

Having introduced here a number of constructions and results to establish the basis of such a connection, several issues remain open, of which some are being addressed at present in our research. Among them, we single out the need for further correspondence results, *i.e.*, how typical notions and results in coalgebra theory (for example, final and weakly final constructions, invariants and assertions) are reflected at the logic level. And, conversely, how structural aspects in the logic, for example *finitarity* (recall a closure operator C is finitary if $C\,X = \bigcup\{C\,Y \mid Y \subseteq X \wedge Y \text{ finite}\}$) are captured and analyzed at the coalgebraic level. Interpolation properties, that can be defined in terms of theories, are interesting candidates [12]. Lifting the entire framework to a more structured setting, where logics are considered over algebras with non empty signature, is part of our current work. This entails the need for capturing logics as dialgebras [39]: the algebraic component models the underlying algebra, while the coalgebraic one expresses consequence.

As a main conclusion we would like to emphasize that the present paper contributes to the recent, on-going research effort to apply methods and

results from Abstract Algebraic Logic to Computer Science problems. Other examples in a similar direction, from our own work, include the semantics of object-oriented programming through hidden k-logics introduced in [31], and the theory of hidden k-state machines as a unified model of specifications expressed in different logical paradigms [27, 28] and in the context of classical automata theory [13]. We hope this approach to program refinement through interpretations, and the associated coalgebraic machinery discussed here, will prove equally fruitful in the near future.

Acknowledgements. The authors acknowledge the financial support by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project `FCOMP-01-0124-FEDER-010047` (project MONDRIAN), as well as within the Center for Research & Development in Mathematics and Applications (CIDMA) of Universidade de Aveiro under the project `PEst-C/MAT/UI4106/2011` with COMPETE number `FCOMP-01-0124-FEDER-022690`. M. Martins was further supported by the project *Nociones de Completud*, reference `FFI2009-09345` (Spain), and A. Madeira by `SFRH/BDE/33650/2009`, a PhD grant jointly supported by FCT and *Critical Software S.A., Portugal*.

Finally, the authors would like to thank an anonymous referee who carefully read a previous version of the paper and made several useful suggestions to improve the paper.

References

[1] M. Abadi and L. Cardelli. *A Theory of Objects*. Springer-Verlag, 1996.

[2] J. Adamek. An introduction to coalgebra. *Theory and Applications of Categories*, 14(8):157–199, 2005.

[3] L. S. Barbosa, J. N. Oliveira, and A. M. Silva. Calculating invariants as coreflexive bisimulations. In J. Meseguer and G. Rosu, editors, *Algebraic Methodology and Software Technology, 12th International Conference, AMAST 2008, Urbana, IL, USA, July 28-31, 2008, Proceedings*, pages 83–99. Springer Lect. Notes Comp. Sci. (5140), 2008.

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

[5] R. Bird and O. Moor. *The Algebra of Programming*. Series in Computer Science. Prentice-Hall International, 1997.

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

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

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

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

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

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

[12] J. Czelakowski and D. Pigozzi. Amalgamation and interpolation in abstract algebraic logic. In X. Caicedo and C. H. Montenegro, editors, *Models, Algebras, and Proofs*, pages 187–265. Lecture Notes in Pure and Applied Mathematics (vol. 203), 1998.

[13] L. Descalço, A. Madeira, and M. A. Martins. Applying abstract algebraic logic to classic automata theory: an exercise. In F. Ferreira, Guerra H., and E. Mayordomo, editors, *Programs, Proofs and Processes; Computability in Europe Cie 2010*, pages 146–157, 2010.

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

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

[16] J.M. Font and R Jansana. *A general Algebraic Semantics for Sentential Logics , 2nd edition*, volume 7. Lecture Notes in Logic, 2009.

[17] P. J. Freyd and A. Ščedrov. *Categories, Allegories*, volume 39 of *Mathematical Library*. North-Holland, 1990.

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

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

[20] C. Hermida and B. Jacobs. Structural induction and coinduction in a fibrational setting. *Information and Computation*, 145:105–121, 1998.

[21] A. Kock. Strong functors and monoidal monads. *Archiv für Mathematik*, 23:113–120, 1972.

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

[23] D. Lucanu, E. Goriac, G. Caltais, and G. Rosu. Circ: A behavioral verification tool based on circular coinduction. In *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings*, pages 433–442. Springer Lect. Notes Comp. Sci. (5728), 2009.

[24] R. D. Maddux. The origin of relation algebras in the development and axiomatization of the calculus of relations. *Studia Logica*, 50(3-4):42–455, 1991.

[25] Clare E. Martin, Sharon A. Curtis, and Ingrid Rewitzky. Modelling angelic and demonic nondeterminism with multirelations. *Sci. Comput. Program.*, 65(2):140–158,

2007.

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

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

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

[29] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation. In Dang Van Hung and Padmanabhan Krishnan, editors, *7th IEEE International Conference on Software Engineering and Formal Methods (SEFM'09)*, pages 250–259. IEEE Computer Society Press, 2009.

[30] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement by interpretation in a general setting. In J. Derrick E. Boiten and S. Reeves, editors, *Proc. Refinement Workshop 2009, Electr. Notes Theor. Comput. Sci. (256)*, pages 105–121. Elsevier, 2009.

[31] M. A. Martins and D. Pigozzi. Behavioural reasoning for conditional equations. *Math. Struct. Comput. Sci.*, 17(5):1075–1113, 2007.

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

[33] Sun Meng and L. S. Barbosa. Components as coalgebras: The refinement dimension. *Theor. Comp. Sci.*, 351:276–294, 2005.

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

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

[36] 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.

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

[38] D. Park. Concurrency and automata on infinite sequences. pages 561–572. Springer Lect. Notes Comp. Sci. (104), 1981.

[39] Erik Poll and Jan Zwanenburg. From algebras and coalgebras to dialgebras. In H. Reichel, editor, *Coalgebraic Methods in Computer Science (CMCS'2001)*, number 44 in ENTCS. Elsevier, 2001.

[40] V. Pratt. Origins of the calculus of binary relations. In *Proc. IEEE Symp. on Logic in Computer Science, Santa Cruz, CA,USA*, pages 248–254. IEEE, 1992.

[41] 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.

[42] E. P. Robinson. Variations on algebra: monadicity and generalisations of equational theories. *Formal Aspects of Computing*, 13:308–326, 2002.

[43] J. Rutten. Universal coalgebra: A theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. (Revised version of CWI Techn. Rep. CS-R9652, 1996).

[44] D. Sannella and A. Tarlecki. Towards Formal Development of Programs from Algebraic Specifications: Implementations Revisited. *Acta Informatica*, 25(3):233–281, 1988.

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

[46] J. Da Silva, I. 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), 1968.

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

[48] A. Tarski. On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89, 1941.

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

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

Manuel A. Martins
CIDMA
Dep. Mathematics
Universidade de Aveiro
Aveiro, Portugal
martins@ua.pt

Alexandre Madeira
Critical Software
MAP-i Doctoral Programme
Aveiro, Portugal
madeira@ua.pt

Luis S. Barbosa
HASLab \ INESC TEC
Universidade do Minho
Braga, Portugal
lsb@di.uminho.pt