Abstract Algebraic Logic and Computer Science

Jacinta Poças[1], Manuel Martins[2], and Carlos Caleiro[1,3]

[1] Dep. Mathematics, IST - TU Lisbon, Portugal
[2] Dep. Mathematics, U Aveiro, Portugal
[3] SQIG - Instituto de Telecomunicações

Abstract. Abstract algebraic logic (AAL) is a branch of logic that uses universal algebra to study the properties of logical systems by associating them with representative classes of algebras, thus generalizing the Lindenbaum-Tarski process that leads to linking Boolean algebras with classical propositional logic. The theory of AAL then classifies logical systems and their metaproperties, systematically, along bridge theorems that relate them with properties of the associated algebras.
Recently, concepts of behavioral algebraic specification have influenced the development of the behavioral approach to AAL. Namely, the notion of behavioral equivalence, imported from computer science, has been used to weaken the traditional equational reasoning underlying AAL, thus accommodating, in a unified theory, a wider range of logical systems. In the opposite direction, the theory of AAL has found meaningful applications in computer science, namely in the study of specification refinement, a central process in modular software development.
Herein, we wish to support the idea that this mutual feedback should deserve to be given closer attention in undergraduate curricula in logic and computer science, thus providing a common solid algebraic background to students wishing to pursue their studies in either area.

1 Introduction

In this paper, we discuss the recent development of the behavioral approach to AAL that adds more relevance to its interdisciplinary nature and provides strong supporting arguments to its inclusion in undergraduate curricula in logic and computer science. Typically, students begin studying logic by having contact with particular logical systems such as classical propositional logic (CPL) and first-order logic (FOL), but sometimes also intuitionistic propositional logic (IPL) or even modal logic. However, deeper studies require a more abstract and systematic approach, which may well be provided by the AAL setting.

Often, logical systems are presented by axioms and inference rules, but this definition is restricted to logics which are finitary. A *logic* (or a *deductive system*), in Tarski´s point of view, consists of a pair $\mathcal{L} = \langle \Sigma, \vdash \rangle$, where Σ is a signature (defining the similarity type of the operations with which formulas are built) and $\vdash$ is a relation between sets of formulas and individual formulas, called the *consequence relation* of $\mathcal{L}$, which satisfies reflexivity, cut, weakening and structurality conditions (cf. [20]), but which may not be finitary.

On the other hand, it is enlightening to associate a class of algebras to a logic. For instance, to CPL we can associate the class of Boolean algebras (BA) as follows: the set of formulas $\mathrm{Fm}_{\mathcal{L}}$ is partitioned into logical equivalence classes and then abstracted by the familiar algebraic process of forming quotients. This is called the *Lindenbaum-Tarski algebra.* More precisely, given a theory T, the Lindenbaum-Tarski algebra induced by T for CPL is the quotient algebra $\mathrm{Fm}_{\mathcal{L}}/\equiv_T$, where $\equiv_T$ is the congruence on the formula algebra defined by $\varphi \equiv_T \psi$ iff φ and ψ are logically equivalent in T, that is, $\varphi \leftrightarrow \psi \in T$. This quotient algebra is a BA. Conversely, every countable BA is isomorphic to an algebra $\mathrm{Fm}_{\mathcal{L}}/\equiv_T$ for some theory T of CPL. This is called the *Lindenbaum-Tarski process.* A similar phenomenon occurs with IPL and the class of Heyting algebras (HA).

In both cases above, there is a biconditional $\leftrightarrow$ that defines the logical equivalence. Still, many logics o not have a proper biconditional, and hence the Lindenbaum-Tarski process cannot be applied directly. In order to generalize this process to other logics, the role played by the congruence $\equiv_T$ is replaced by the Leibniz congruence and the equivalence connective by a system of equivalence formulas [3]. A *system of equivalence formulas* is a set of formulas in two variables that satisfies reflexivity, symmetry, transitivity, modus ponens and simple replacement conditions. The *Leibniz congruence* $\Omega(T)$ on the term algebra over a $\mathcal{L}$-theory T is defined as the relation that identifies two formulas α, β if for every formula φ and any variable p occurring in φ, $\varphi(p/\alpha) \in T$ iff $\varphi(p/\beta) \in T$. The Leibniz congruence is extended in a natural way to the power set of an arbitrary algebra. Given an algebra $\mathbf{A}$, of the same similarity type as $\mathcal{L}$, and a designated subset F of A, the pair $\langle \mathbf{A}, F\rangle$ is called a *matrix.* The relation $\Omega(F)$ identifies any two elements which cannot be distinguished by any property expressed by a formula; formally, for any pair of elements $a, b \in A$; $a \equiv b\,(\Omega(F))$ if for each formula $\varphi(x, \overline{y})$, where $\overline{y}$ is a vector of k variables, and all parameters $\overline{c} \in A^k$; $\varphi^{\mathbf{A}}(a, \overline{c}) \in F$ iff $\varphi^{\mathbf{A}}(b, \overline{c}) \in F$. Moreover, $\Omega(F)$ is a congruence on $\mathbf{A}$. When it is the identity relation, the matrix $\langle \mathbf{A}, F\rangle$ is called *reduced.* The terminology is justified by the fact that $\Omega(F)$ may be seen as the sentential version of the second order definition of equality given by Leibniz, who considered two objects equal if they share the same properties expressed in the language of discourse.

We can gather logics in classes by means of properties of the Leibniz operator, the so-called *Leibniz Hierarchy.* The class of algebraizable logics ([3]) is a very important class in this hierarchy. A logic $\mathcal{L} = \langle \Sigma, \vdash \rangle$ is said to be *algebraizable* iff there exists a class K of algebras such that the equational consequence relation $\models_K$ is equivalent to $\vdash$. This link between logic and universal algebra is very powerful because it relates two areas of mathematics. Moreover, for algebraizable logic $\mathcal{L}$, we can relate logical properties of $\mathcal{L}$ with algebraic properties of its algebraic counterpart. This kind of results has been called *bridge theorems.* There are many examples of such theorems; for instance: an algebraizable logic has Craig´s interpolation property iff the class of its reducts of its reduced matrix models has the amalgamation property [1, 11].

The development of the behavioral approach to AAL [9] was inspired by techniques from behavioral specification in computer science. Namely, the weaker notion of algebraization is based on behavioral equivalence, and allows for extending the applicability of theory to logics which are not algebraizable according to the standard approach, while also bringing a new algebraic perspective to logics which are algebraizable using the standard tools of AAL. On the other hand, a generalization of the notion of sentential logic, hidden k-logic, has been used to support verification and specification of requirements in the object oriented paradigm. Naturally, its theory has been developed based on tools and results from AAL. Namely, the Leibniz hierarchy is generalized for hidden k-logics using a behavioral version of the Leibniz congruence [9, 15]. This feedback between AAL and computer science is worthy of further research, and we are confident that this correlation will exert, in the near future, positive developments in both sides. Furthermore, as of now, we view as highly positive the possibility of exploring this interdisciplinarity as a vehicle for teaching logic and its connections with computer science in a solid algebraic environment.

Outline. The paper is organized in two parts. In the first part, Section 2, we present the behavioral generalization of the notion of algebraizable logic. Namely, we show how some well known arguments/ideas from computer science are used in the development of the behavioral approach to AAL, and present a simple example. At the end of this section, we describe the class of behaviorally finitely equivalential logics and we exhibit an element in the class. In the second part, Section 3, we explain how AAL theory can be applied to computer science in the context of software verification and development. It is well known that the traditional notions of refinement based on signature morphisms are often too rigid to capture a number of relevant transformations, hence other maps should be considered. We explain how interpretations can be used for this purpose.

2 From computer science to AAL

The motivation for the *behavorial* approach to AAL emerges from computer science, namely from the algebraic approach to the specification and verification of software, where abstract data types and object classes are defined by the properties of their associated operations. In that setting, data can naturally be split into two categories: visible data which can be directly accessed, and hidden data that can only be accessed indirectly by analyzing the meaning of programs with visible output, called *experiments*. The role of experiments is to access the relevant information encapsulated in a state. Since we cannot access the hidden data directly, it is not possible to reason about the equality of two hidden values. Hence, equational logic needs to be replaced by behavioral equational logic based on the notion of behavioral equivalence. Two values are said to *behaviorally equivalent* if they cannot be distinguished by the set of available experiments. Since we may not have all experiments available to distinguish two values, we introduce the notion of Γ-behavioral equivalence, where Γ is a subset of the set of original operations. Two values are said to be Γ-*behaviorally equivalent* if they cannot be distinguished by all experiments that can be build with the operations in Γ. The Γ-behavioral equivalence is the largest Γ-congruence whose visible part is the identity relation. Thus there is a natural connection with the notion of Leibniz congruence. The standard notion of algebraization corresponds to the particular case when $\Gamma = \Sigma$.

The behavioral generalization of the standard AAL theory is motivated by the fact that simple logics are often not expressive enough when we want to reason about complex systems. In particular, we often need logics over richer languages whose elements can be distinguished by sorts. For instance, FOL can be naturally seen as a logic with two sorts (terms and formulas). In the resulting behavioral setting, many logics that lack a meaningful algebraic counterpart in the traditional sense become behaviorally algebraizable [13, 9, 6]. The approach is supported on replacing the role of unsorted equational logic by many-sorted behavioral equational logic over the same signature and taking as unique visible sort, the sort ϕ of formulas. Since the sort ϕ is considered visible, we have equational reasoning about formulas, but we do not require every connective to be congruent. Paradigmatic examples are the paraconsistent logic C_1 of da Costa and the Carnap-style presentation of modal logic S_5 which are not algebraizable in the standard sense but they are behaviorally algebraizable [13, 9, 7]. In a very informal way, a logic $\mathcal{L}$ is *algebraizable* if there exists a strong representation between $\mathcal{L}$ and the equational consequence associated with a class K of algebras.

A simple example ([8]). Let us consider the logic $\mathcal{K}/2$ from [2], built over a signature with negation $\neg$ and implication $\Rightarrow$. Its consequence relation $\vdash_{\mathcal{K}/2}$ can be simply defined to be the semantic entailment associated to the set of all bivaluations $v : \mathrm{Fm} \to \{0, 1\}$, with 1 designated, such that:

- $v(\neg\varphi) = 0$ if $v(\varphi) = 1$, and
- $v(\varphi \Rightarrow \psi) = 0$ iff $v(\varphi) = 1$ and $v(\psi) = 0$.

Clearly, the logic would be classical if the first clause would be written with 'iff'. As it is, it has a classical implication but a paracomplete negation. Interestingly, the derived unary operation $\sim$, with $\sim\varphi$ defined as an abbreviation of $\varphi\Rightarrow(\neg\varphi)$, still behaves as a classical negation: $v(\sim\varphi) = 0$ iff $v(\varphi) = 1$.

$\mathcal{K}/2$ is structural, finitary and very easily axiomatizable by: Still, $\mathcal{K}/2$ is

(A1) $\vdash_{\mathcal{K}/2} A \Rightarrow (B \Rightarrow A)$
(A2) $\vdash_{\mathcal{K}/2} (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
(A3) $\vdash_{\mathcal{K}/2} ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
(A4) $\vdash_{\mathcal{K}/2} A \Rightarrow ((\neg A) \Rightarrow B)$
(MP) $A, A \Rightarrow B \vdash_{\mathcal{K}/2} B$

not algebraizable in the traditional sense. However, the logic is $\mathit{\Gamma}$-behaviorally algebraizable with $\mathit{\Gamma}$ consisting of $\Rightarrow$ and $\sim$, equivalence set $\{\varphi \Rightarrow \psi, \psi \Rightarrow \varphi\}$ and truth-defining set $\{\varphi \approx (\varphi \Rightarrow \varphi)\}$.

The class of behaviorally finitely equivalential logics. The development of the behavioral approach to AAL opens plenty new lines of research. For instance, we can study the behavioral version of the Leibniz hierarchy [9]. An interesting class of logics in the hierarchy is the class of behaviorally finitely equivalential logic. A logic $\mathcal{L}$ is Γ-*behaviorally finitely equivalential* if there exists a finite set Δ of formulas in two variables that satisfies reflexivity, symmetry, transitivity, modus ponens and simple replacement, but only for operations in Γ. The set Δ is called a Γ-*behavioral finite equivalence set* for $\mathcal{L}$. We can also characterize the class of behaviorally finitely equivalential logics by properties of the *behavioral Leibniz operator* which maps each theory T of $\mathcal{L}$ to the largest Γ-congruence over the algebra of formulas compatible with T. Indeed, assume that a many-sorted logic $\mathcal{L}$ is Γ-*standard* (for every sort there exists a formula without variables). Then, we have that, $\mathcal{L}$ is Γ-behaviorally finitely equivalential iff the behavioral Leibniz operator is monotone and continuous.

This new class is distinct from others already studied. Actually, there is an example of a logic which is only equivalential in the standard sense but finitely equivalential in the behavioral approach: normal modal logic K [14]. Namely, it is equivalential with the set of equivalence formulas $\{\Box^n(\varphi \leftrightarrow \psi) : n \in \mathbb{N}\}$ but not finitely equivalential in the standard sense. As usual $\Box^n\alpha$ is an abbreviation for the formula $\Box\Box\cdots\Box\alpha$, in which the modal operator $\Box$ appears n times. However, normal modal logic K is $\mathit{\Gamma}$-behaviorally finitely equivalential, if we exclude the modal operator from $\mathit{\Gamma}$, with the finite set of behavioral equivalence formulas $\{\varphi \to \psi, \psi \to \varphi\}$.

3 From AAL to computer science

The industrial need for high-assurance of critical systems requires mathematically based development methods, able to model complex systems at ever-increasing levels of reliability and security. As a consequence, the interest of mathematicians in computer science problems has been steadily growing. An example is the development of logical systems to support verification and specification of requirements. Hidden k-logics have been used, as the underlying logic, in the verification and specification theory of object oriented systems [18]. Hidden k-logics are a natural generalization of logics in Tarski's sense, so its theory has been developed within AAL. More recently, the notion of refinement has been examined using more flexible logical interpretations, as the traditional notion of refinement based on signature morphisms is often too rigid. The process of stepwise refinement, through which a complex design is produced by incrementally adding details and reducing non determinism with respect to its original, high-level specification, is done step-by-step until the specification becomes a precise description of a concrete model; technically, an algebra.

A *specification* $SP = \langle \Sigma, [\![SP]\!] \rangle$ is formed by a signature Σ, denoted by $\mathrm{Sig}(SP)$ and a class of Σ-algebras $[\![SP]\!]$. The *setpwise refinement process* is the systematic procedure by which, from an initial abstract specification SP_0 more concrete specifications are built by introducing new requirements leading to a chain of specifications

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

where for all $1 \leqslant i \leqslant n$, $SP_{i-1} \rightsquigarrow SP_i$ means $[\![SP_i]\!] \subseteq [\![SP_{i-1}]\!]$. Composition assures that $SP_0 \rightsquigarrow SP_n$. The classical tool to relate specifications over distinct signatures is the signature morphism. A *signature morphism* connecting $\Sigma = (S, \Omega)$ to $\Sigma' = (S', \Omega')$ is a pair of maps $\sigma = (\sigma_{sorts}, \sigma_{op})$, where $\sigma_{sorts} : S \to S'$ and $\sigma_{op} : \Omega \to \Omega'$ is a $(S^* \times S)$-indexed family of functions respecting the sorts of operations. Given a signature morphism $\sigma : \Sigma \to \Sigma'$, we say that the specification SP' over Σ' is a *σ-refinement of SP*, in symbols $SP \rightsquigarrow_\sigma SP'$, if $[\![SP']\!]\restriction_\sigma \subseteq [\![SP']\!]$, where $[\![SP']\!]\restriction_\sigma = \{\mathbf{A}\restriction_\sigma \mid \mathbf{A} \in [\![SP']\!]\}$. We should point out that a signature morphism maps a term into just another term and must be compatible with the operations, a notion too rigid to be useful in general. In order to capture more features of the development of software systems, we can search for transformations supported by mappings which need not to be morphisms; e.g. multi-functions. Unfortunately, in general it is not any longer possible to define the reduct of an algebra by these maps, and consequently the traditional algebraic treatment of program development can not be adopted.

A possibility is to use *interpretations*. An *interpretation* is a multi-function that translates a formula into a set of formulas by preserving meaning. Actually, originally defined as a tool for studying equivalent algebraic semantics (c.f. [3, 5]), the notion of interpretation proves effective to capture a number of transformations difficult to deal with in classical terms. To illustrate, let us consider the following two specifications:

spec SPEC1 =
sorts s
ops $f : s \to s$
axioms
$t \approx t$
inference rules

$$\frac{t \approx t', t' \approx t''}{t \approx t''} \qquad \frac{t \approx t'}{t' \approx t} \qquad \frac{t \approx t'}{f(t) \approx f(t')}$$

spec SPEC2 =
sorts s
ops $ok : \to s$
$f : s \to s$
$test : s \times s \to s$
axioms
$test(t, t) \approx ok$
inference rules

$$\frac{test(t, t') \approx ok, test(t', t'') \approx ok}{test(t, t'') \approx ok} \qquad \frac{test(t, t') \approx ok}{test(t', t) \approx ok} \qquad \frac{test(t, t') \approx ok}{test(f(t), f(t')) \approx ok}$$

Since the axiomatization of SPEC2 is defined by the translation of the inference rules and axioms of SPEC1, we have $\vdash_{\text{SPEC1}} t \approx t'$ iff $\vdash_{\text{SPEC2}} test(t, t') \approx ok$. This shows that SPEC2 interprets SPEC1 by the schematic multifunction $f(x \approx y) = \{test(x, y) \approx ok\}$. 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 $\iota : \text{Sig(SPEC1)} \to \text{Sig(SPEC2)}$, which obviously does not witness a refinement step between them.

We are now ready to formulate this new tool for formal development of programs. A *translation* from a signature $\Sigma = (S, \Omega)$ to a signature $\Sigma' = (S', \Omega')$ with respect to (w.r.t.) the set of variables X and X' (for Σ and Σ' resp.) is a $S - S'$-sorted multi-function $\tau : \mathrm{Eq}(\Sigma, X) \multimap \mathrm{Eq}(\Sigma', X')$ such that the image of each equation is a finite set. Let $\tau : \mathrm{Eq}(\Sigma, X) \multimap \mathrm{Eq}(\Sigma', X')$ be a translation and SP be a specification over Σ. We say that τ *interprets* SP if there is a specification SP' over Σ' such that, for any Σ-conditional equation ξ over X, $SP \models \xi$ iff $SP' \models \tau(\xi)$. We say that a specification SP' over Σ' *refines the specification* SP *via the interpretation* τ, in symbols $SP \rightharpoondown_{\tau} SP'$, if τ interprets SP and for any Σ-conditional equation ξ over X, $SP \models \xi \Rightarrow SP' \models \tau(\xi)$ (c.f., [17]). A paradigmatic example is the class HA, that can be regarded as a refinement of the specification of BA. Let X be a set of variables and Σ the usual signature for BA and for HA. Consider the double negation map: $\iota : T_{\Sigma}(X) \to T_{\Sigma}(X)$ such that $\iota(t) = \neg\neg t$.

Let τ be the self translation of Σ defined by $\tau(t \approx t') = \{\iota(t) \approx \iota(t')\}$. It can be shown that τ interprets the specification of BA in the specification of HA [5].

The notions of σ-refinement and refinement by interpretation are strongly related. Indeed, let SP be a specification over Σ and τ a translation from Σ to Σ' w.r.t. the set of variables X and X' which interprets SP. Then, for every SP' specification over Σ', if $SP^{\tau} \rightsquigarrow SP'$ then $SP \rightharpoondown_{\tau} SP'$, where SP^{τ} is the specification over Σ' whose models are the τ-model class of SP, i.e., the Σ'-algebras $\mathbf{A}'$ such that $SP \models \xi$ implies $\mathbf{A}' \models \tau(\xi)$ for any Σ-conditional equation ξ over X. Actually, the refinement by interpretation is a generalization of σ-refinement when the signature morphism σ is injective. The notion of refinement by interpretations is generalized, in a natural way, to k-logics, that is k-dimensional logics (cf. [17]). Equational logic is the most interesting example of k-logics by considering an equation $t \approx t'$ as a pair $\langle t, t' \rangle$. This generalization allows the possibility to move from one dimension to another one. For instance, any subclass of the class of BA induces a refinement by interpretation of CPL with τ the usual translation defined by $\tau(\varphi) = \{\langle \varphi, \top \rangle\}$ (see [17]).

The AAL tools has also been applied to automata theory by considering bisimulation as a special case of the Leibniz congruence ([4], [16] and [12]). We are confident that such approach can be generalized to general Kripke models.

4 Conclusion

We briefly presented AAL as a theory that studies the mechanism by which a class of algebras can be associated with a logic, and provides a general context in which bridge theorems relating metalogical properties of a logic to algebraic properties of its algebraic counterpart can be formulated precisely. We also discussed how ideas from behavioral algebraic specification have influenced the development of the behavioral approach to AAL. In this respect, we introduced the new class of behaviorally finitely equivalential logics which captures some phenomena occurring within modal logics. We also pointed out that AAL can be applied to concrete problems in computer science, namely in algebraic program development. To conclude, we want to emphasize that, beyond its genesis, these new links with computer science increase its interdisciplinary nature and reassert the pertinence of considering AAL as a vehicle for studying logic.

Acknowledgments. This research was partially supported by FCT under the project PTDC/EIA-CCO/108302/2008 Mondrian. The second author also acknowledges partial financial assistance by the project "Nociones de Completud", reference FFI2009-09345 (MICINN - Spain).

References

1. Andréka, H., Németi, I., Sain, I.: Algebraic logic. in D. M. Gabbay and F. Guenther, (eds.). Handbook of Philosophical Logic, Dordrecht, Kluwer Academic Publishers. **2** (2001) 133–247.
2. Béziau, J.-Y.: Classical Negation can be Expressed by One of its Halves. Logic Journal of the IGPL. (1999) 145–151.
3. Blok, W.J., Pigozzi, D.: Algebraizable logics. Memoirs of the American Mathematical Society. **396** (1989).
4. Blok, W.J. and Pigozzi, D.:Algebraic semantics for universal Horn logic without equality. Universal algebra and quasigroup theory, Lect. Conf., Jadwisin/Pol. 1989, Res. Expo. Math. 19 (1992) 1-56.
5. Blok, W.J., Regbagliato, J.: Algebraic semantics for deductive systems. Studia Logica. **74** (2003) 153–180.
6. Caleiro, C., Gonçalves, R.: On the algebraization of many-sorted logics. in J. Fiadeiro and P.Y. Schobbens, (eds.), Recent Trends in Algebraic Development Techniques - Selected Papers. in Lecture Notes in Computer Science. **4409** (2007) 21–36.
7. Caleiro, C., Gonçalves, R.: Behavioral algebraization of da Costa's C-systems. Journal of Applied Non-Classical Logics. **19(2)** (2009) 127–148.
8. Caleiro, C., Gonçalves, R.: Abstract valuation semantics. Submitted for publication.
9. Caleiro, C., Gonçalves, R., Martins, M.A.: Behavioral algebraization of logics. in Studia Logica. **91** (2009) 63–111.
10. Czelakowski, J.: *Protoalgebraic Logics.* Trends in logic, Studia Logica, Kluwer Academic Publishers (2001).
11. Czelakowski, J., Pigozzi, D.: Amalgamation and interpolation in abstract algebraic logic. in X. Caicedo and C.H. Montenegro, (eds.), Models, algebras and proofs. in Lecture Notes in Pure Appl. Math. **203** (1999) 187–265.
12. 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*, (2010) 146–157.
13. Gonçalves, R.: Behavioral algebraization of logics. PhD Thesis. Technical University of Lisbon, Instituto Superior Técnico (2008).
14. J. Malinowski. Equivalence in intensional logic, IFIS PAN (1990). Available at `http://www.home.umk.pl/~jacekm/arlint.pdf`.
15. Martins, M.A.: Closure properties for the class of behavioral models. *Theor. Comput. Sci.*, **379** (2007) 53–83.
16. M. A. Martins. On the behavioral equivalence between k-data structures. *Comp. J.*, 51(2) (2008) 181–191.
17. Martins, M.A., Madeira, A. and Barbosa, L.S.: Refinement by interpretation in a general setting. In J. Derrick E. Boiten and S. Reeves, (eds.). *Proc. Refinement Workshop,* Elsevier (2009) 105–121.
18. Martins, M.A. and Pigozzi, D.: Behavioural reasoning for conditional equations. *Math. Struct. in Comput. Sci.* **17** (2007) 1075–1113.
19. Poças, J.: Leibniz Hierarchy. MSc Thesis. University of Aveiro (2009).
20. Tarski, A.: Logic, semantics, and metamathematics (2nd Edition). Hackett Pub Co., Indianapolis, Indian (1983).