An algebraic way to unequally treat equal values

Marc Aiguier[1 2] , Gilles Bernot[1] , Pascale Le Gall[1]

mail: {aiguier,bernot,legall}@univ-evry.fr

[1] L.I.V.E., Département Mathématiques–Informatique, Université Evry - Val d'Essonne, Bd. des Coquibus, F-91025 Evry cedex, France
Tel: (33.1) 69 47 71 44 and Fax: (33.1) 69 47 70 08

[2] L.R.I., UA CNRS 410, Université Paris-Sud, Bât. 490, F-91405 Orsay cedex, France

Draft version July, 7th, 1993

Abstract

Surprisingly, in the field of abstract data types, it proves useful to specify some properties that do not propagate through equalities. The recent framework of *label algebras* is devoted to this problem. We prove the syntax and semantics as well as the main results of label algebras; then we define a calculus, the *label calculus*, and we prove its completeness with respect to the semantics of label specifications.

Keywords: algebraic specifications, theorem proving, complete calculus, initial semantics.

1 Introduction

Talking about the correctness of a software requires at first to rigorously answer the question "correct with respect to what ?" This implies to provide a *formal specification* of what it is supposed to do. On the contrary, talking about the correctness of a formal specification is a non sense[1] because "what we have mind" is not formally defined (and such a definition is precisely the specification). It is only possible to partially verify a specification by stating certain properties that should be satisfied if the specification does what it is supposed to do. Thus, it is of first interest to provide the specifier with a complete calculus.

One of the main advantages of algebraic semantics for formal specifications is to provide

[1]But the problem is simplified as a specification is considerably smaller than a program.

the specifier with powerful specification building primitives. In this paper we present the recent theory of *label algebras*([BL91], [BL93], [LeG93]).

The main "exoticism" of our framework is to give up the Leibniz law: for some properties $\mathcal{P}$, the deduction rule $\frac{\mathcal{P}(x)\ ,\ x=y}{\mathcal{P}(y)}$ is not correct with respect to the semantics of label algebras. Such a particularity is useful for exception handling purposes or observability issues for instance. One of the simplest formal specification where the Leibniz law is not desirable is the case of "bounded natural numbers with recoveries." Any programming language with exception handling (such as CLU, ADA, CAML...) can implement, for example, the interval $[0..\mathit{Maxint}]$ with the two following properties:

1. if the operation *successor* is applied to the bound *Maxint* then it raises the exception *TooLarge* (definition of an exceptional case);
2. if the operation *successor* raises the exception *TooLarge* then do not perform it, and everything goes on as if *successor* had never been applied (exception handler).

From the algebraic specification point of view, if the expression "$t \epsilon \mathit{TooLarge}$" signifies "$t$ raises the exception *TooLarge*", then these two properties are respectively expressed as follows:

(1) $$\mathit{succ}(\mathit{Maxint})\ \epsilon\ \mathit{TooLarge}$$

(2) $$\mathit{succ}(n)\ \epsilon\ \mathit{TooLarge} \Longrightarrow \mathit{succ}(n) = n$$

(within the framework of label algebras, " ϵ " will be read as "*is labelled by*".)

Let *BeforeLast* be the predecessor of *Maxint* in the interval $[0..\mathit{Maxint}]$. Clearly, we have $\mathit{succ}(\mathit{BeforeLast}) = \mathit{Maxint}$ and we do not want for this application of *succ* to be exceptional (it should not raise *TooLarge*), else we would have, from the second axiom, $\mathit{succ}(\mathit{BeforeLast}) = \mathit{BeforeLast}$ (and the exception handling would be performed "too early").
Moreover, we have $\mathit{succ}(\mathit{Maxint}) = \mathit{Maxint}$ (via the last axiom, n being substituted by *Maxint*; this is usually called a *recovery* within the terminology of exception handling).
Let $\mathcal{P}(x)$ be the property defined as "$x\ \epsilon\ \mathit{TooLarge}$". This property does not follow the Leibniz law: we have $\mathcal{P}(\mathit{succ}(\mathit{Maxint}))$ and $\mathit{succ}(\mathit{Maxint}) = \mathit{succ}(\mathit{BeforeLast})$ but we do not have $\mathcal{P}(\mathit{succ}(\mathit{BeforeLast}))$.

Indeed, the *terms* $\mathit{succ}(\mathit{Maxint})$ and $\mathit{succ}(\mathit{BeforeLast})$ share the same *value* but they do not share the same computation. The term $\mathit{succ}(\mathit{Maxint})$ is exceptional whilst $\mathit{succ}(\mathit{BeforeLast})$ is not. The semantics of label algebras involves both terms and values unlike all other algebraic semantics which uniquely consider values. The main contribution of our framework is to allow the specifier to talk about values and computations leading to these values, using the same language. For example the expression "$succ(Maxint)$" in "$succ(Maxint) = Maxint$" denotes a value whilst, in "$succ(Maxint) \ \epsilon \ TooLarge$", it denotes a computation.

A good formal specification theory should provide us with (at least): syntax, semantics, modularity aspects and theorem proving facilities. Sections 2 and 3 show that the theory of label algebras fulfills the three first criterions. In Section 4 we introduce a complete calculus according to its semantics: the *label calculus*. Lastly, a short example shows, using the inference rules, some useful aspects of label algebras and label calculus (Section 5).

We assume that the reader is familiar with the elementary concepts of category theory [McL71], algebraic specifications [GTW78][EM85] and the main outlines of the Birkhoff's completeness proof [Bir35].

2 Label algebras

Reminders : ("Classical ADJ approach," [GTW78][EM85])

- A signature is a couple $\Sigma = \langle S, F\rangle$ where S is a finite set of sorts (or type names) and F is a finite set of operation names with arity in S. A Σ-algebra is a heterogeneous set, A, partitioned as $A = \{A_s\}_{s\in S}$, and with, for each operation name "$f : s_1 \cdots s_n \to s$" in F (with $0 \leq n$), a total function $f_A : A_{s_1} \times \cdots \times A_{s_n} \to A_s$. The Σ-morphisms are obviously the sort preserving, operation preserving applications.
- Given a heterogeneous set of variables $V = \{V_s\}_{s\in S}$, the free Σ-term algebra with variables in V is the least Σ-algebra $T_\Sigma(V)$ (with respect to the preorder induced by the Σ-morphisms) such that $V \subseteq T_\Sigma(V)$.
- Since V is not necessarily finite or enumerable, we can consider $T_\Sigma(A)$ for every algebra A. An element of $T_\Sigma(A)$ is a Σ-term such that each leaf contains either a constant of Σ, or a value of A. Moreover, in this case, every term of $T_\Sigma(A)$ can be canonically evaluated on the algebra A via the so-called evaluation morphism $eval_A : T_\Sigma(A) \to A$. The Σ-morphism $eval_A$ relates each term to its final value.

Since label algebras carry informations on both terms and values (cf. Section 1), a simple idea could be to use the initial morphism from T_Σ (the ground term algebra over Σ) to A. Unfortunately, this morphism is not always surjective. The main technical point underlying the framework of label algebras is to systematically use the surjective morphism $eval_A$ from $T_\Sigma(A)$ to A. This allows us to treat terms with non reachable values in order to cope with enrichment, parametrization or abstract implementation.

Notation 1 : We note $\overline{A} = T_\Sigma(A)$ and for every Σ-morphism $\mu : A \to B$, we note $\overline{\mu} : \overline{A} \to \overline{B}$ the canonical Σ-morphism that extends μ to the corresponding free algebras. When there is no ambiguity, we shall still denote μ the morphism $eval_B \circ \overline{\mu}$ from $\overline{A}$ to B.

Definition 2 : A *label signature* is a couple $\Sigma L = \langle \Sigma, L \rangle$ where $\Sigma = \langle S, F \rangle$ is an usual signature and L is a finite set (of "*labels*").

For example, *TooLarge* would be a label in order to specify bounded natural number.

Definition 3 : Given a label signature ΣL, a *ΣL-algebra* (or *label algebra*) is a couple $\mathcal{A} = (A, \{l_A\}_{l \in L})$ where A is a Σ-algebra, and $\{l_A\}_{l \in L}$ is a L-indexed family such that, for each l in L, l_A is a subset of $\overline{A}$.

There are no conditions about the subsets l_A: they can intersect several sorts, they are not necessarily disjoint and their union $(\bigcup_{l \in L} l_A)$ does not necessarily cover $\overline{A}$.

Example 4 : Let $S = \{Nat\}$, $F = \{zero : Nat \;,\; succ : Nat \to Nat\}$ and $L = \{TooLarge\}$. A label algebra $\mathcal{A} = (A, \{TooLarge_A\})$ can be defined on this signature as follows:

- A is the interval $[0..Maxint]$ of $I\!N$, with $zero_A = 0$, $succ_A(i) = i + 1$ for $i \in [0..Maxint[$ and $succ_A(Maxint) = Maxint$.
  Then $\overline{A} = \{\ succ^i(a) \mid i \in I\!N,\ a \in A \cup \{zero\}\ \}$
  ($succ^i(a)$ stands for $succ(succ(\cdots(succ(a))..))$, the operation $succ$ being written i times).
- $TooLarge_A = \{succ^i(a) \mid i \in I\!N,\ a \in A \cup \{zero\},\ i + eval_A(a) = Maxint + 1\}$.

Definition 5 : Let $\mathcal{A} = (A, \{l_A\}_{l \in L})$ and $\mathcal{B} = (B, \{l_B\}_{l \in L})$ be two ΣL-algebras. A ΣL-morphism (or *label morphism*) $h : \mathcal{A} \to \mathcal{B}$ is a Σ-morphism from A to B such that $\forall l \in L,\ \overline{h}(l_A) \subseteq l_B$.

The category of all ΣL-algebras and ΣL-morphisms is denoted by $Alg(\Sigma L)$.

Notation 6 : Let ΣL be a label signature.

- Given a set of variables V, $\mathcal{T}_{\Sigma L}(V)$ is the ΣL-algebra such that the underlying Σ-algebra is the term algebra $T_\Sigma(V)$ and for each l in L, $l\_T_\Sigma(V)$ is empty.
- $\mathcal{T}_{\Sigma L}$ is defined by $\mathcal{T}_{\Sigma L} = \mathcal{T}_{\Sigma L}(\emptyset)$ and is called the ground term ΣL-algebra.

The ΣL-algebra $\mathcal{T}_{\Sigma L}$ is clearly initial in $Alg(\Sigma L)$.

Definition 7 : Let $\Sigma L = \langle \Sigma, L \rangle$ be a label signature.

- An *atom* is either an equality $(u = v)$ such that u and v are Σ-terms with variables, u and v belonging to the same sort, or a labelling atom $(w \mathrel{\epsilon} l)$ such that w is a Σ-term with variables and l belongs to L.
  "$(w \mathrel{\epsilon} l)$" should be read "w is labelled by l".
- A positive conditional ΣL-axiom (or *label axiom*) is a formula of the form

  $$\alpha_1 \wedge \cdots \wedge \alpha_n \Longrightarrow \alpha$$

  where the α_i and α are (positive) atoms.

In the remainder of this paper, label formulas (or shortly axioms) will always be positive conditional ones[2].

The satisfaction relation is the most important definition of this section: notice that we consider assignments with range in $\overline{A} = T_\Sigma(A)$ (terms) instead of A (values).

Definition 8 : Let $\mathcal{A} = (A, \{l_A\}_{l \in L})$ be a ΣL-algebra.

- $\mathcal{A}$ satisfies $(u = v)$, where u and v are two terms of the same sort in $\overline{A}$, means that $eval_A(u) = eval_A(v)$ [the last symbol "$=$" being the set-theoretic equality in the carrier of A].
- $\mathcal{A}$ satisfies $(w \mathrel{\epsilon} l)$, where $w \in \overline{A}$ and $l \in L$, means that $w \in l_A$ [the symbol "$\in$" being the set-theoretic membership].
- Let φ be a label axiom of the form $\alpha_1 \wedge \cdots \wedge \alpha_n \Longrightarrow \alpha$. $\mathcal{A}$ satisfies φ, denoted by $\mathcal{A} \models \varphi$, means that for all assignments $\sigma : V \to \overline{A}$ (V covering all the variables of φ), if $\mathcal{A}$ satisfies $\sigma(\alpha_i)$ for all $i = 1..n$ (according to the "ground atomic satisfaction" defined above) then $\mathcal{A}$ also satisfies $\sigma(\alpha)$.

$\mathcal{A}$ satisfies a label specification $SP = \langle \Sigma L, Ax \rangle$ (where Ax is a finite set of ΣL-axioms) if and only if $\mathcal{A}$ satisfies all the axioms of Ax. $Alg(SP)$ (sometimes denoted $Alg(Ax)$) is the corresponding full subcategory of $Alg(\Sigma L)$.

Example 9 : The label algebra described in Example 4 satisfies the axioms (1) and (2) p. 2 (where "*Maxint*" is a notation for $succ^{Maxint}(0)$ in the axiom (1)).

[2]One can consider every well formed formulas using any usual connectives and quantifiers, the semantics of which is defined in [LeG93]. However the label calculus defined here is complete only in the positive conditional framework.

3 Main results and applications

As usual when considering positive conditional axioms, the main results are initiality results (least congruence, initial SP-algebra, left adjoint to the forgetful functor...). In the field of abstract data types, such properties provide the specifier with one of the simplest way to express modularity in an initial approach [GTW78][EM85][Ber87]. These results are not proved here; complete proofs can be found in [BL92].

Definition 10 : Let $\Sigma L = \langle \Sigma, L \rangle$ be a label signature. Let $\mathcal{A} = (A, \{l_A\}_{l \in L})$ be a ΣL-algebra.

- A *ΣL-relation* (or *label relation*) on $\mathcal{A}$ is a couple $\mathcal{R} = (R, \{l_R\}_{l \in L})$ where R is a binary relation on A compatible with the sorts and $\{l_R\}_{l \in L}$ is a family of subsets of $\overline{A}$.
- A ΣL-congruence (or *label congruence*) is a ΣL-relation $\Theta = (\equiv_\Theta, \{l_\Theta\}_{l \in L})$ such that $\equiv_\Theta$ is a usual Σ-congruence on A and $l_A \subseteq l\_\Theta$ for each l in L.

Proposition 11 : Let $\mathcal{A} = (A, \{l_A\}_{l \in L})$ be a ΣL-algebra and let $\Theta = (\equiv_\Theta, \{l_\Theta\}_{l \in L})$ be a ΣL-congruence. Let $A/\_\Theta$ be the usual quotient Σ-algebra of A by the Σ-congruence $\equiv_\Theta$ and $q : A \to A/\_\Theta$ the corresponding quotient Σ-morphism. Let $\{l_{A/\_\Theta}\}_{l \in L}$ be defined by $l_{A/\_\Theta} = \overline{q}(l\_\Theta)$ for each l in L.
The couple $(A/\_\Theta, \{l_{A/\_\Theta}\}_{l \in L})$ is a ΣL-algebra, denoted by $\mathcal{A}/\_\Theta$, and q is a label morphism. This label algebra is called the *quotient algebra* of $\mathcal{A}$ by Θ.

Theorem 12 : *(Fundamental theorem for initiality issues)*
Let $\mathcal{A}$ be a label algebra. Let $\mathcal{R}$ be any label relation on $\mathcal{A}$. There exists a least label congruence Θ on $\mathcal{A}$ such that $\mathcal{R} \subseteq \Theta$ (i.e. $R \subseteq \equiv_\Theta$ and $\forall\, l \in L$, $l_R \subseteq l\_\Theta$).

Corollary 13 : Let SP be a label specification. The category $Alg(SP)$ has an initial object $\mathcal{T}\_SP$.

Another corollary, for structured specifications, is that the forgetful functor has a left adjoint functor [LeG93]; however the framework of label algebras does not form a liberal institution [GB84].

As already mentioned, the peculiar role of labels is to give up the Leibniz law: two terms with equal values can nonetheless get different labels. In a way, sorts can be shown as type names on values whilst labels can be shown as type names on terms. Nevertheless, the Leibniz law can be restored with finesse via the axiom

$$x \,\epsilon\, l \;\wedge\; x = y \Longrightarrow y \,\epsilon\, l$$

for each desired label l and each desired sort s (the sort of the variables x and y). Initiality results remain because these "partial Leibniz laws" are positive conditional. This way, labels can be turned into sorts and subsorting can easily be simulated by labels: the fact that l is a subsort of l' can still be expressed by a positive conditional axiom[3]

$$x \epsilon\ l \Longrightarrow x \epsilon\ l'$$

Similarly, several observational semantics of algebraic specifications can be expressed by using a label "*Obs*" that characterizes if a term is observable (as in [Hen89] or [BB91]). Moreover, even if label algebras are total algebras (the semantics of the operations are total functions) we can simulate partial functions via a label that characterizes "defined" terms. Algebraic semantics of exception handling can also be treated via label algebras: a label "*Ok*" characterizes non exceptional terms. For more details on the possible applications of label algebras, see [BL92][BL93][LeG93]. A revealing example based on similar ideas is developed in Section 5.

Lastly, let us remark that label algebras do not treat labels as ordinary values. In particular a label cannot itself be labelled. On the contrary Equational Typed Logic [MSS90], Unified Algebras [Mos89] or G-algebras [Meg90] treat sorts as "first class citizens" (a sort can belong to another sort). We investigate the consequencies of giving up the Leibniz law whilst they investigate extensions of the notion of subsort without loosing the Leibniz law. Indeed, label algebras and those other extensions of order sorted algebras have not the same purpose. Of course, the expressive power of label specifications can be reached in a first order logic framework, but here, we take advantage of a specialization to algebraic semantics (terseness of specifications, already existing specification building primitives, etc.).

4 A complete calculus for label algebras

Definition 14 : Given a label signature $\Sigma L = \langle S, \Sigma, L \rangle$ and a heterogeneous set of variables V, the *label calculus* is defined by the following set of inference rules, where Ax denotes a set of axioms, α and β denote atoms, Γ denotes a finite associative and commutative conjunction[4] of atoms, t, t_i, u_j and v_j denote Σ-terms with variables, $\rho : V \to T_\Sigma(V)$ denotes a substitution and $f : s_1 \cdots s_n \to s$ denotes any operation of Σ, u_j and v_j being of sort s_j.

[3]In particular we retrieve the usual initiality results of order sorted algebras [Gog78][FGJM85].

[4]More precisely, the preconditions of label axioms are considered as finite sets of atoms, the symbol $\wedge$ being the insertion in those sets. This exempt the user from explicitly managing associativity and commutativity rules for the conjunction in the inference steps.

Axiom introduction:

if $(\Gamma \Rightarrow \alpha)$ **is an axiom of** Ax **then** $Ax \vdash (\Gamma \Rightarrow \alpha)$

Tautology:

$Ax \vdash (\alpha \Rightarrow \alpha)$

Monotonicity:

if $Ax \vdash (\Gamma \Rightarrow \alpha)$ **then** $Ax \vdash (\Gamma \wedge \beta \Rightarrow \alpha)$

Modus Ponens

if $Ax \vdash (\Gamma \wedge \beta \Rightarrow \alpha)$ **and** $Ax \vdash (\Gamma \Rightarrow \beta)$ **then** $Ax \vdash (\Gamma \Rightarrow \alpha)$

Reflexivity:

$Ax \vdash t = t$

Symmetry:

if $Ax \vdash (\Gamma \Rightarrow t_1 = t_2)$ **then** $Ax \vdash (\Gamma \Rightarrow t_2 = t_1)$

Transitivity:

if $Ax \vdash (\Gamma \Rightarrow t_1 = t_2)$ **and** $Ax \vdash (\Gamma \Rightarrow t_2 = t_3)$ **then** $Ax \vdash (\Gamma \Rightarrow t_1 = t_3)$

Replacement:

if, $\forall j = [1..n], Ax \vdash (\Gamma \Rightarrow u_j = v_j)$ **then** $Ax \vdash (\ \Gamma \Rightarrow f(u_1..u_n) = f(v_1..v_n)\ )$

Substitution:

if $Ax \vdash (\Gamma \Rightarrow \alpha)$ **then** $Ax \vdash (\rho(\Gamma) \Rightarrow \rho(\alpha))$

We recognize the classical rules of equational reasoning (taking into account positive conditional formulas) except the Leibniz law. Indeed, we saw that the Leibniz law has not to be correct with respect to the label algebra semantics (cf. the algebra $\mathcal{A}$ of the example 4). There is no rule which explicitely concerns labelling, but notice that the rule "*Substitution*" implicitely also concerns label atoms. On the contrary all other algebraic approaches require specific rules to ensure the Leibniz law (e.g. Equational Typed Logic [MSS90]).

In order to prove the completeness of the label calculus we follow a proof similar to the Birkhoff's one [Bir35] (see also [MSS90] for another proof of completeness calculus). In this article, we give the main outlines of the proof (see also [Aig92]).

Definition 15 : Let ΣL be a label signature. Let Ax be a set of label axioms. Let Γ be a finite conjunction of atoms. Let V be a set of variables containing the variables appearing in Ax and Γ. We note $\Theta_\Gamma^{Ax} = (\equiv\_\Gamma, \{l_\Gamma\}_{l \in L})$ the label relation defined on $\mathcal{T}_{\Sigma L}(V)$ by:

$$t \equiv\_\Gamma u \text{ if and only if } Ax \vdash (\Gamma \Rightarrow t = u)$$

and for any term t of $\overline{T_\Sigma(V)}$:

$$t \in l\_\Gamma \text{ if and only if } \exists t' \in T_\Sigma(V), \begin{cases} \exists \rho : V \to T_\Sigma(V), \overline{\rho}(t') = t \\ Ax| - (\Gamma \Rightarrow t' \epsilon\ l) \end{cases}$$

Lemma 16 : Θ_Γ^{Ax} is a label congruence. Moreover, let $\mathcal{T}_\Gamma^{Ax} = \mathcal{T}_{\Sigma L}(V)/\_\Theta_\Gamma^{Ax}$ denote the corresponding quotient algebra; for any atom α we have:

$$[\mathcal{T}_\Gamma^{Ax} \models (\Gamma \Rightarrow \alpha)] \Longrightarrow [Ax \vdash (\Gamma \Rightarrow \alpha)]$$

Proof: The rules *Reflexivity, Symmetry, Transitivity* and *Replacement* of the label calculus ensure that the relation $\equiv\ \_\Gamma$ is an usual congruence relation on the underlying Σ-algebra $T\_\Sigma L(V)$. Moreover, as $l\_T\_\Sigma L(V)$ is empty, then by definition, Θ_Γ^{Ax} is a label congruence.

Let Γ be of the form $\alpha_1 \wedge \cdots \wedge \alpha_n$. From the rules of *Tautology* and *Monotonicity* we get: $\forall i \in [1..n]$, $Ax \vdash (\Gamma \Rightarrow \alpha_i)$ and by definition of Θ_Γ^{Ax}: $\forall i \in [1..n]$, $\mathcal{T}_\Gamma^{Ax} \models \alpha_i$. This last fact allows us to only consider $\mathcal{T}_\Gamma^{Ax} \models \alpha$ when considering $\mathcal{T}_\Gamma^{Ax} \models (\Gamma \Rightarrow \alpha)$.

Let us suppose $\mathcal{T}_\Gamma^{Ax} \models (\Gamma \Rightarrow \alpha)$. We have then by the definition of validation: $\mathcal{T}_\Gamma^{Ax} \models \overline{q} \circ \overline{\rho}(\alpha)$ where ρ is a substitution from V to $T_\Sigma(V)$. We get the two following cases depending on the form of α:

1. Let α be of the form $t = u$. By definition of $\mathcal{T}_\Gamma^{Ax}$ and Θ_Γ^{Ax}, we get $t \equiv\ \_\Gamma u$ and $Ax \vdash \Gamma \Rightarrow t = u$.
2. Let α be of the form $t\ \epsilon\ l$. For the same reasons, $Ax \vdash (\Gamma \Rightarrow t\ \epsilon\ l)$.

Finally, one can conclude that: $\mathcal{T}_\Gamma^{Ax} \models [(\Gamma \Rightarrow \alpha)] \Longrightarrow [Ax \vdash (\Gamma \Rightarrow \alpha)]$. □

Lemma 17 : For all Γ, $\mathcal{T}_\Gamma^{Ax}$ satisfies Ax.

Proof: Let $\alpha_1 \wedge \cdots \wedge \alpha_m \Rightarrow \alpha$ be in Ax (we note Γ' the conjunction $\alpha_1 \wedge \cdots \wedge \alpha_m$) and let $\sigma : V \to \overline{\mathcal{T}_\Gamma^{Ax}}$ be an assignment such that $\mathcal{T}_\Gamma^{Ax} \models \sigma(\Gamma')$. We have to prove that: $\mathcal{T}_\Gamma^{Ax} \models \sigma(\alpha)$.

From σ, we can construct by induction a substitution called $\rho : V \to T_\Sigma(V)$ such that: $\sigma = \overline{q} \circ \overline{\rho}$.

From now on, at each time we will consider an atomic formulae, two cases will appear depending on the form of this considered atomic formulae. Let us suppose that α is written

$t = u$ (resp. $t \epsilon l$). From the equality $\sigma = \overline{q} \circ \overline{\rho}$, it suffices to prove that $\rho(t) \equiv \_\Gamma \rho(u)$ (resp. $\overline{\rho}(t) \in l\_\Gamma$).

We successively obtain the following steps taking into account the generic definition of the congruence Θ_Γ^{Ax}:

- $t \equiv \_\Gamma' u$ (resp. $t \in l\_\Gamma'$) by the rule *Axiom introduction*;
- $\rho(t) \equiv \_\rho(\Gamma')\rho(u)$ (resp. $\overline{\rho}(t) \in l\_\rho(\Gamma')$) by the rule *Substitution*;
- $\rho(t) \equiv \_(\Gamma \wedge \rho(\Gamma'))\rho(u)$ (resp. $\overline{\rho}(t) \in l\_(\Gamma \wedge \rho(\Gamma'))$) by the rule *Monotonicity*;
- for all j in $[1..m]$, α_j is written either $u_j = v_j$ or $u_j \epsilon l_j$. By the hypothesis $(\mathcal{T}_\Gamma^{Ax} \models \sigma(\Gamma'))$, we get respectively either $\rho(u_j) \equiv \_\Gamma \rho(v_j)$ or $\overline{\rho}(u_j) \in l\_\Gamma$;
- $\rho(t) \equiv \_\Gamma \rho(u)$ (resp. $\overline{\rho}(t) \in l\_\Gamma$) by the rule *Modus Ponens*.

Therefore, we obtain that for all Γ, $\mathcal{T}_\Gamma^{Ax}$ satisfies Ax. □

Theorem 18 : *(Completeness of the label calculus)*

Let Ax be a set of label axioms. Let φ be any formula. We have:

$$[Ax \vdash \varphi] \Longrightarrow [\forall \mathcal{A} \in Alg(Ax), \mathcal{A} \models \varphi]$$

Proof: Let $(\Gamma \Rightarrow \alpha)$ be the formula φ.

The left to right implication of the theorem is the *soundness* of the label calculus. The proof is based on an induction on the proof lenght by means of an decomposition depending on the nature of the last rule applied. We only mention below the case of the *Substitution* rule since we have already pointed out that it is the most specific rule to treat labelling.

We suppose that the rule *Substitution* is the last applied rule in the proof.

Let $\Gamma \Rightarrow \alpha$ be a formulae and let $\rho : V \to T_\Sigma(V)$ be a substitution such that $Ax \vdash (\rho(\Gamma) \Rightarrow \rho(\alpha))$ is the last step of our proof.

We state the induction hypothesis that: $\forall \mathcal{A} \in Alg(Ax), \mathcal{A} \models (\Gamma \Rightarrow \alpha)$. By definition of the validation, it means: $\forall \sigma : V \to \overline{A}$, $\mathcal{A} \models \sigma(\Gamma) \Rightarrow \sigma(\alpha)$.

By induction hypothesis, we have: $\forall \sigma' : V \to \overline{A}, \mathcal{A} \models \sigma'(\rho(\Gamma)) \Rightarrow \sigma'(\rho(\alpha))$ because $\sigma' \circ \rho$ is a particular case of the substitutions σ. Therefore, by the definition of the validation, we get: $\mathcal{A} \models \rho(\Gamma) \Rightarrow \rho(\alpha)$.

The reverse implication results from the two above lemmas: let us assume that

$$\forall \mathcal{A} \in Alg(Ax), \mathcal{A} \models \varphi$$

Since φ is written $(\Gamma \Rightarrow \alpha)$ and since $\mathcal{T}_\Gamma^{Ax}$ satisfies Ax by Lemma 17, we have $\mathcal{T}_\Gamma^{Ax} \models \varphi$; thus, from Lemma 16, $\quad Ax \vdash \varphi$. $\square$

5 An example

Among many other possible applications, label algebras allow us to specify evaluation strategies (e.g. "bottom-up" evaluation, also called the "call-by-value" strategy). We would like to point out that this is obtained *without giving up the advantages of the algebraic approach*: well chosen label atoms in the preconditions of the axioms allow the specifier to properly restrict pattern matching. This is entirely due to the absence of the Leibniz law.

Let us consider a simple example: natural numbers with the usual generators 0 and *succ_* and the defined operation *pred_*. Our goal in this example is to restrict the scope of the equations $(pred(succ(n)) = n)$ and $(succ(pred(n)) = n)$ in order to reflect only a bottom-up evaluation. We will use a label *BottomUp*. The expression "$t \ \epsilon \ BottomUp$" will mean that the term t can be evaluated by call-by-value via these two equations.

Of course, the constant operation 0 and all terms of the form $succ(\cdots succ(0)..)$ "get a value" via the bottom-up evaluation (0 and *succ* being free generators). This can be reflected by the two following axioms

$$0 \ \epsilon \ BottomUp \tag{3}$$

$$n \ \epsilon \ BottomUp \implies succ(n) \ \epsilon \ BottomUp \tag{4}$$

In order to restrict the scope of the equation $(pred(succ(n)) = n)$ to a bottom-up evaluation, it is sufficient to state that a term of the form $pred(succ(n))$ gets a value via this equation if the subterm "$succ(n)$" and the right hand side "n" of the equation previously get a value. This gives rise to the following axiom

$$n \ \epsilon \ BottomUp \ \wedge \ succ(n) \ \epsilon \ BottomUp \implies pred(succ(n)) = n \tag{5}$$

Similarly, the equation $(succ(pred(n)) = n)$ becomes

$$n \ \epsilon \ BottomUp \ \wedge \ pred(n) \ \epsilon \ BottomUp \implies succ(pred(n)) = n \tag{6}$$

Let us remark that axioms (3)(4)(5) and (6) do *not* imply, for example, that the term $pred(succ(0))$ is labelled by *BottomUp*, even though $pred(succ(0)) = 0$ and $0 \ \epsilon \ BottomUp$ are deductible. Nevertheless, in order to continue the bottom-up evaluations, we have to propagate the label *BottomUp*. We can obtain this propagation by recopying the preconditions of axioms (5) and (6) as follows

(7) $\quad n \in BottomUp \ \wedge \ succ(n) \in BottomUp \ \Longrightarrow \ pred(succ(n)) \in BottomUp$

(8) $\quad n \in BottomUp \ \wedge \ pred(n) \in BottomUp \ \Longrightarrow \ succ(pred(n)) \in BottomUp$

Then, using the label calculus with axioms (3) to (8), we prove for example that $pred(succ(0)) \in BottomUp$ and $pred(succ(0)) = 0$ as follows:

[*a*]

$$0 \in BottomUp$$

[from axiom (3)]

[*b*]

$$0 \in BottomUp \Rightarrow succ(0) \in BottomUp$$

[from axiom (4) and Substitution with $n \leftarrow 0$]

[*c*]

$$0 \in BottomUp \ \wedge \ succ(0) \in BottomUp \Rightarrow pred(succ(0)) \in BottomUp$$

[from axiom (7) and Substitution with $n \leftarrow 0$]

[*d*]

$$0 \in BottomUp \Rightarrow pred(succ(0)) \in BottomUp$$

[from Modus Ponens applied to [*b*] and [*c*]]

[*e*]

$$pred(succ(0)) \in BottomUp$$

[from Modus Ponens applied to [*a*] and [*d*]]

[*f*]

$$0 \in BottomUp \ \wedge \ succ(0) \in BottomUp \Rightarrow pred(succ(0)) = 0$$

[from axiom (5) and Substitution with $n \leftarrow 0$]

[*g*]

$$0 \in BottomUp \Rightarrow pred(succ(0)) = 0$$

[from Modus Ponens applied to [b] and [f]]

[h]

$$pred(succ(0)) = 0$$

[from Modus Ponens applied to [a] and [g]]

On the contrary, it is impossible to prove $succ(pred(0)) = \langle AnythingElse \rangle$ because there is no axiom whose conclusion is of the form $t \mathrel{\epsilon} BottomUp$ such that t matches $pred(0)$. Since the label calculus is complete (Theorem 18) this implies that $succ(pred(0))$ is not a "standard" natural number[5] in the initial algebra of this specification.

The reader might believe that axioms (7) and (8) could be replaced by a simpler axiom reflecting the Leibniz law:

(9) $$n = m \;\wedge\; n \mathrel{\epsilon} BottomUp \;\Longrightarrow\; m \mathrel{\epsilon} BottomUp$$

It does *not* work. As a matter of fact, let us consider the axiom

(10) $$pred(0) = 0$$

From one hand let us consider Axiom (10) together with axioms (3) to (8) and let us consider the term $succ(pred(0))$. Still, the subterm $pred(0)$ cannot be proved labelled by $BottomUp$; consequently only axiom (10) can be applied and the "*Replacement*" rule of label calculus shows that $succ(pred(0)) = succ(0)$. The point is that, without axiom (9), $pred(0)$ is not labelled by $BottomUp$ although it is proved equal to 0, consequently axiom (6) cannot be applied and $succ(pred(0))$ is *not* equal to 0 in the initial algebra.
From another hand, if (7) and (8) are replaced by (9), then $pred(0)$ is labelled by $BottomUp$ (as it is equal to 0). In this case, axiom (6) can be applied and we finally get that $succ(0) = 0$ leading to a trivial initial algebra.

6 Conclusion

We have described the syntax and semantics of *label specifications* (in the positive conditional case). The main particularity of label algebras is to give up the Leibniz law (replacement of equal by equal) while preserving the advantages of the algebraic approach (modularity issues, software engineering applications, etc.).

We have defined the *label calculus*. The label calculus is complete with respect to the label algebra semantics: Section 4 explains how to adapt the Birkhoff's completeness proof ([Bir35]) to label calculus.

[5]i.e. it is not in the class of a term of the form 0 or $succ(\cdots(succ(0)..)$.

An detailed example, developed in Section 5, illustrates the main advantage of label algebras. Roughly speaking: to provide the specifier with a refined "control of pattern matching" within an algebraic framework.

References

[Aig92] Aiguier M. : "*Un calcul pour les algèbres étiquetées.*" DEA Report, University of Orsay, Paris XI, France, Sept. 1992.

[BB91] Bernot G., Bidoit M. : "*Proving the correctness of algebraically specified software: Modularity and Observability issues.*" Proc. of AMAST-2, Second Conference of Algebraic Methodology and Software Technology, Iowa City, Iowa, USA, May 1991.

[BL91] Bernot G., Le Gall P. : "*Label algebras : a systematic use of terms.*" 8th International Workshop on Abstract Data Types, Dourdan, August 1991. LNCS 655, p.144-163, Springer-Verlag, 1993.

[BL92] Bernot G., Le Gall P. : "*Label algebras and exception handling.*" Draft version, also in Habilitation Thesis of Bernot G., University of Orsay, Paris XI, France, Feb. 1992.

[BL93] Bernot G., Le Gall P. : "*Exception handling and term labelling.*" Proc. of TAPSOFT'93 (FASE), Orsay, France, 13-17 Apr. 1993. LNCS 668, p.421-436, Springer-Verlag, 1993.

[Ber87] Bernot G. : "*Good functors . . . are those preserving philosophy !*" Proc. Summer Conference on Category Theory and Computer Science, September 1987, Springer-Verlag LNCS 283, p.182-195.

[Bir35] Birkhoff G. : "*On the structure of abstract algebras.*" Proc. of the Cambridge Philosophical Soc. 31, pp. 433-454, 1935.

[EM85] Ehrig H., Mahr B. : "*Fundamentals of Algebraic Specification 1. Equations and initial semantics.*" EATCS Monographs on Theoretical Computer Science, Vol.6, Springer-Verlag, 1985.

[FGJM85] Futatsugi K., Goguen J., Jouannaud J-P., Meseguer J. : "*Principles of OBJ2.*" Proc. 12th ACM Symp. on Principle of Programming Languages, New Orleans, January 1985.

[GB84] Goguen J.A., Burstall R.M. : "*Introducing institutions.*" Proc. of the Workshop on Logics of Programming, Springer-Verlag L.N.C.S. 164, pp.221-256, 1984.

[GTW78] Goguen J.A., Thatcher J.W., Wagner E.G. : "*An Initial Algebra Approach to the Specification, Correctness, and Implementation of Abstract Data Types.*" Current Trends in Programming Methodology, ed. R.T. Yeh, Printice-Hall, Vol.IV, pp.80-149, 1978.

[Gog78] Goguen J.A. : "*Order sorted algebras: exceptions and error sorts, coercion and overloading operators.*" Univ. California Los Angeles, Semantics Theory of Computation Report n.14, Dec. 1978.

[Hen89] Hennicker R. : "*Implementation of Parameterized Observational Specifications.*" TapSoft, Barcelona, LNCS 351, vol.1, pp.290-305, 1989.

[LeG93] Le Gall P. : "*Les algèbres étiquetées : une sémantique pour les spécifications algébriques fondée sur une utilisation systématique des termes. Application au test de logiciel avec traitement d'exceptions.*" PhD Thesis, University of Orsay, France, Jan. 1993.

[MSS90] Manca V., Salibra A., Scollo G. : "*Equational Typed Logic.*" Theoretical Computer Science, Vol.77, p.131-149, 1990.

[McL71] Mac Lane S. : "*Categories for the working mathematician.*" Graduate texts in mathematics, 5, Springer-Verlag, 1971.

[Meg90] Mégrelis A. : "*Algèbre galactique - Un procédé de calcul formel, relatif aux semifonctions, à l'inclusion et à l'égalité.*" Ph.D. Thesis, University of Nancy I, Sept. 1990.

[Mos89] Mosses P. : "*Unified algebras and Institutions.*" Proc. of IEEE LICS'89, Fourth Annual Symposium on Logic in Computer Science, June 1989, Asilomar, California.