Behavioural Constructor Implementation for Regular Algebras

Sławomir Lasota*

Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warszawa, Poland
sl@mimuw.edu.pl

Abstract. We investigate *regular algebras*, admitting infinitary regular terms interpreted as least upper bounds of suitable approximation chains. The main result of this paper is an adaptation of the concept of behavioural constructor implementation, studied widely e.g. for standard algebras, to the setting of regular algebras. We formulate moreover a condition that makes proof of correctness of an implementation step tractable. In particular, we indicate when it is sufficient to consider only finitary observational contexts in the proofs of behavioural properties of regular algebras.

Keywords: Algebraic specifications, observational equivalence, regular algebras, behavioural constructor implementation, proofs of behavioural properties.

Introduction

Behavioural semantics of algebraic specifications is widely accepted to capture properly the "black box" character of data abstraction. As a nontrivial example of an algebraic framework where behavioural ideas may be applied, we consider regular algebras, differing from the standard algebras in one respect: they allow one to additionally model "infinite" datatypes, like streams. Regular algebras were introduced in [16], and then studied e.g. in [17, 6]. Our starting point here is a more recent paper [4], investigating observational equivalence of regular algebras and the induced behavioural semantics of specification. Regular algebras contain properly continuous algebras [15], intended usually to model "infinite" datatypes. Unfortunately continuous algebras are not well suited for behavioural semantics of specification, as they lack some crucial algebraic properties, e.g. quotients of continuous algebras do not compose (cf. [5]).

The subject of this paper is to analyze the applicability of regular algebras as models of behavioural specifications in the process of stepwise development of software systems. A general methodology is proposed for regular algebras, as an adaptation of the *constructor behavioural implementation* [12, 13]. To our knowledge the constructor implementation has not been studied in this setting so

* The work reported here was partially supported by the KBN grant 8 T11C 019 19.

far. Whereas regular algebras require a separate treatment, since their structure differs substantially from standard algebras: they have partially ordered carrier sets and all term-definable mappings have fixed points, given by the least upper bounds of appropriate approximation chains.

This paper reports briefly contents of Chapter 5 of [9]. After preliminary Section 1 introducing regular algebras, observational equivalence and constructor implementation, in Section 2 we explain how to combine, roughly, regular and standard algebras. We extend the setting of regular algebras with a possibility to have carriers of some sorts essentially unordered (these sorts are called *algebraic*), which is strongly required in practical examples.

In Section 3 we prove that observational indistinguishability in a regular algebra is induced by only finitary observable contexts – surprisingly, even when non-trivial approximation chains exist in carrier sets of observable sorts. This makes proofs of behavioural properties of regular algebras substantially easier, and allows one to exploit methods known for standard algebras, like context induction [7] or results of [1, 2]. Moreover, when all observable sorts are algebraic, the notion of observational equivalence can be characterized by only finitary contexts too. Hence one can prove equivalence of regular algebras e.g. using standard observational correspondences [14].

Finally, in Sections 4 and 5 we propose an adaptation of behavioural constructor implementation methodology [12, 13] to the new framework. Initially, our main motivation was to enable finitary implementations of infinitary regular data structures. To this aim we introduce a method, called μ-induction, for defining fixed-points of recursors. This gives rise, at an intermediate step in constructor implementation, to *pre-regular algebras*, defined as algebras of a syntactical monad. Next we investigate proofs of correctness of such an implementation step. A property is formulated, called *behavioural consistency*, that guarantees that such proofs are tractable. In Lemma 1 and Theorem 3 we a derive sufficient condition for this property to hold. The condition formalizes a methodological paradigm to ensure behavioural consistency: when defining a pre-regular algebra in the implementation step, one should ensure that $\perp$ in each sort has the smallest observable behaviour and the observable behaviour of each least upper bound of an approximation chain is the least upper bound of the observable behaviours of approximants. Moreover, under that condition, the results from Section 3 apply and one only needs to consider finitary observational contexts while proving correctness of implementation. Our general considerations are illustrated by few examples in Section 1 and in Appendix.

Acknowledgements The author is grateful to Andrzej Tarlecki for many fruitful discussions and valuable comments during this work.

1 Preliminaries

Regular algebras Let Σ, Σ' be fixed many-sorted algebraic signatures throughout this paper. We omit introducing classical notions of standard Σ-algebra, homomorphism (the category of those is denoted Alg_Σ), subalgebra, congruence, quotient (cf. e.g. [5]). By $t_{A[v]}$ we denote the value of Σ-term t in algebra A under valuation v. For A being a standard or regular algebra, by $|A|$ we denote the many-sorted carrier set of A; by $|A|_s$ the carrier of sort s; by $|A|_S$, for a subset S of sorts of Σ, the carrier sets of sorts from S. All sets are implicitly meant to be many-sorted in the sequel.

An *ordered Σ-algebra* is a standard Σ-algebra whose carrier set on each sort s is partially ordered (let $\leq_s^A$ denote below the partial order in $|A|_s$) and has a distinguished element $\perp_s^A \in |A|_s$. However, we do not assume operations in an ordered algebra to be monotonic.

The set of *regular Σ-terms* $T_\Sigma^\mu(X)$ over X is defined inductively as usual, with the only additional case: for any $t \in T_\Sigma^\mu(X \cup \{z : s\})_s$ and a distinguished variable z of the same sort s, there is a *μ-term* $\mu z.t$ in $T_\Sigma^\mu(X)_s$. Similarly, the inductive definition of the value $t_{A[v]}$ of a term under a valuation $v : X \to |A|$ in an ordered algebra A needs one more case. For $t \in T_\Sigma^\mu(X \cup \{z : s\})_s$, put:

- $t^0_{A[v]}(\perp) = \perp_s^A$,
- for $i \in \omega$, $t^{i+1}_{A[v]}(\perp) = t_{A[v_i]}$, where $v_i : X \cup \{z : s\} \to |A|$ extends v by $v_i(z) = t^i_{A[v]}(\perp)$.

Now, $(\mu z.t)_{A[v]}$ is defined if $t^i_{A[v]}(\perp)$ are defined, for all $i \in \omega$, $t^i_{A[v]}(\perp) \leq_s^A t^{i+1}_{A[v]}(\perp)$, and the least upper bound $\bigsqcup_{i\in\omega} t^i_{A[v]}(\perp)$ exists in $|A|_s$; if so, then $(\mu z.t)_{A[v]} = \bigsqcup_{i\in\omega} t^i_{A[v]}(\perp)$. An ordered Σ-algebra A is *regular* if it satisfies the following conditions:

- *completeness*: for all $t \in T_\Sigma^\mu(X)$ and $v : X \to |A|$, the value $t_{A[v]}$ is defined,
- *continuity*: for all $t \in T_\Sigma(X \cup \{y : s\})_{s'}$, $q \in T_\Sigma^\mu(X \cup \{z : s\})_s$ and valuation $v : X \to |A|$, $t_{A[v_i]} \leq_s^A t_{A[v_{i+1}]}$, for $i \in \omega$, and $t_{A[v']} = \bigsqcup_{i\geq 0} t_{A[v_i]}$, where valuation $v' : X \cup \{y : s\} \to |A|$ extends v by $v'(y) = (\mu z.q)_{A[v]}$ and $v_i : X \cup \{y : s\} \to |A|$ extends v by $v_i(y) = q^i_{A[v]}(\perp)$, for $i \geq 0$.

Continuity is required only for *finitary* terms $t \in T_\Sigma(X \cup \{y : s\})_{s'}$, i.e., those not containing symbol μ. A seemingly stronger condition (considered in [4]), concerning all $t \in T_\Sigma^\mu(X \cup \{y : s\})_{s'}$, is equivalent to the above one – a detailed proof can be found in [9]. By completeness, $\perp_s^A$ is the least element in $|A|_s$.

Note that completeness does not imply general ω-completeness. Completeness and continuity as defined above, correspond to ω-completeness and ω-continuity w.r.t. definable ω-chains only. Moreover, operations in a regular algebra need not even be monotonic.

By a regular Σ-homomorphism $h : A \to B$ we mean any function h such that for all terms $t \in T_\Sigma^\mu(X)$ and valuations $v : X \to |A|$, $h(t_{A[v]}) = t_{B[h\circ v]}$. Regular algebras together with regular homomorphisms form a category, called $RAlg_\Sigma$ in the sequel. A regular subalgebra of A is any regular algebra B the carrier of which is a subset of A and such that for all terms $t \in T_\Sigma^\mu(X)$ and valuations $v : X \to |B|$, $t_{B[v]} = t_{A[v]}$. It is easy to see that all operations of B are restrictions of operations of A to $|B|$ and moreover $\perp_s^B = \perp_s^A$ for each sort s. For $Y \subseteq |A|$, by *the subalgebra of A generated by Y* we mean the least regular subalgebra of A whose carrier includes Y.

A relation $\sim \subseteq |A| \times |A|$ is a regular congruence iff $\sim = \sqsubseteq \cap \sqsubseteq^{-1}$, for some *pre-congruence* $\sqsubseteq$, where a pre-congruence is any pre-order on $|A|$ satisfying:

1. for all $t \in T_\Sigma(X \cup \{y : s\})_{s'}$, $q \in T^\mu_\Sigma(X \cup \{z : s\})_s$ and $v : X \to |A|$, the family $\{t_{A[v_i]}\}_{i \geq 0}$ is a chain w.r.t. $\sqsubseteq$ with a least upper bound $t_{A[v']}$, where valuation $v' : X \cup \{y : s\} \to |A|$ extends v by $v'(y) = (\mu z.q)_{A[v]}$ and valuations $v_i : X \cup \{y : s\} \to |A|$ extend v by $v_i(z) = q^i_{A[v]}(\bot)$, for $i \geq 0$; [1]
2. the equivalence $\sqsubseteq \cap \sqsubseteq^{-1}$ is preserved by the operations (i.e., is a standard congruence).

In particular, instantiating t in 1 with a single variable, we get: for all $q \in T^\mu_\Sigma(X \cup \{z : s\})_s$ and $v : X \to |A|$, $\{q^i_{A[v]}(\bot)\}_{i \geq 0}$ is a chain w.r.t. $\sqsubseteq$ with the least upper bound $(\mu z.q)_{A[v]}$. Given a regular congruence $\sim$, the quotient regular algebra $A/\sim$ is defined (for each sort s) by $|A/\sim|_s = |A|_s/_\sim$, $\bot^{A/\sim}_s = [\bot^A_s]$, $[a]_\sim \leq^{A/\sim}_s [a']_\sim \Leftrightarrow a \sqsubseteq a'$ (where $\sqsubseteq$ is a pre-congruence inducing $\sim$) and $f_{A/\sim}([a_1]_\sim, \ldots, [a_n]_\sim) = [f_A(a_1, \ldots, a_n)]_\sim$. For details we refer to [4].

Observational equivalence There have been a number of different formalizations of the concept of behavioural equivalence of algebras (see [8] for an overview). In the following we concentrate on an observational equivalence induced by a subset of observable sorts; hence throughout this paper let us fix a subset OBS of *observable sorts* of Σ.

In the following, let X denote some OBS-sorted set of variables. By a *Σ-context* of sort s' on sort s we mean any term $\gamma \in T^\mu_\Sigma(X \cup \{z_s : s\})_{s'}$, where z_s is a special, distinguished variable of sort s such that $z_s \notin X$. Note that z_s, for $s \notin OBS$, is the only variable of a non-observable sort appearing in a context. A special role is played by *observable contexts*, i.e., contexts of observable sort ($s' \in OBS$). For any regular Σ-algebra A, Σ-context γ on sort s, valuation $v : X \to |A|_{OBS}$ and value $a \in |A|_s$, we will write $\gamma_{A[v]}(a)$ for $\gamma_{A[v_a]}$ where v_a extends v by $v(z_s) = a$.

For any A, let A_{OBS} denote its subalgebra generated by (carrier sets of) observable sorts; we call A_{OBS} the *observational subalgebra* of A. The regular congruence $\sim^{OBS}_A$ on A_{OBS} is defined as follows: for any $a, a' \in |A_{OBS}|_s$, $a \sim^{OBS}_A a'$ if and only if for all valuations v into carriers of observable sorts of A_{OBS} and all observable contexts γ, $\gamma_{A[v]}(a) = \gamma_{A[v]}(a')$. The congruence $\sim^{OBS}_A$ is called *observational indistinguishability* in A; $\sim^{OBS}_A$ is the greatest congruence on A_{OBS} being identity on observable sorts (cf. [4]). The quotient of A_{OBS} by $\sim^{OBS}_A$ represents the observable behaviour of A; $A/\sim^{OBS}$ is *fully abstract* in the sense that its indistinguishability is identity (in particular it equals its own observational subalgebra, i.e., $(A/\sim^{OBS})_{OBS} = A/\sim^{OBS}$). Two regular algebras are taken as equivalent when their behaviours are isomorphic:

[1] In [4] a stronger requirement was assumed, for all regular terms $t \in T^\mu_\Sigma(X \cup \{y : s\})_{s'}$, similarly as in continuity condition above. In the same vein as above, finitary terms are sufficient here.

Definition 1 ([4]). *Observational equivalence* $\equiv_{OBS}$ *of regular* Σ*-algebras is defined by:*

$$A \equiv_{OBS} B \quad \textit{iff} \quad A_{OBS}/_{\sim_A^{OBS}} \cong B_{OBS}/_{\sim_B^{OBS}}.$$

$A_{OBS}/{\sim_A^{OBS}}$ will be written shortly $A/{\sim_A^{OBS}}$ or $A/{\sim^{OBS}}$ in the sequel.

All definitions in this subsection are still valid when standard algebras are taken into account (cf. [3, 2]) – the only modification required is to restrict contexts γ to only finitary terms $T_\Sigma(X)$ in definition of indistinguishability.

Behavioural implementation A concept of observational equivalence plays a crucial role in the process of step-wise refinement. It allows one to consider possibly large class of acceptable realizations of a specification, under the only assumption that the observable behavior of the implementation conforms to the specification requirements. We recall below a formalization of these ideas by the notion of behavioural constructor implementation [12, 13] in the framework of standard algebras.

Let us look at an example specification of stacks of integers, as an illustration of a behavioural approach to implementation:

specification STACKS **extends** INT, BOOL **by**
sorts
$stack$;
operations
$empty : stack$;
$push(\_,\_) : int \times stack \to stack$;
$pop(\_) : stack \to stack$;
$top(\_) : stack \to int$;
$empty?(\_) : stack \to bool$;
axioms
$empty?(empty) = true$,
$empty?(push(n, s)) = false$,
$top(push(n, s)) = n$,
$pop(push(n, s)) = s$.

Consider the following candidate A for implementation of this specification, which realizes a stack as an infinite array of integers (modeled here as a function from natural numbers to integers) together with a pointer to a current position (top of the stack):

$A_{stack} := \mathbb{N} \times \mathbb{Z}^{\mathbb{N}}$,
$empty_A := \langle 0, \lambda i.\ 0\rangle$,
$empty?_A(\langle k, f\rangle) := (k = 0)$,
$push_A(n, \langle k, f\rangle) := \langle k+1, \lambda i.\ \textbf{if}\ \ i = k\ \textbf{then}\ \ n\ \textbf{else}\ \ f(i)\rangle$,
$pop_A(\langle k, f\rangle) := \ \textbf{if}\ \ k = 0\ \textbf{then}\ \ \langle 0, f\rangle\ \textbf{else}\ \ \langle k-1, f\rangle$,
$top_A(\langle k, f\rangle) := \ \textbf{if}\ \ k = 0\ \textbf{then}\ \ 0\ \textbf{else}\ \ f(k-1)$.

Obviously, this algebra does not satisfy the axioms of STACKS; in particular, the last axiom does not hold. On the other hand, intuitively, this seems to be a "reasonable" realization of the datatype of stacks. The intuition behind this is as follows: although $pop(push(n, s))$ and s need not be identical, they are indistinguishable w.r.t. the observable sorts $\{bool, int\}$. This leads to the behavioural semantics of specification, according to which models of a specification $SP = \langle \Sigma, Ax\rangle$ are all algebras which *behaviourally (observationally) satisfy* its axioms:

$$BehMod(SP) = \{A \in Alg_\Sigma : A \vDash_{OBS} Ax\}.$$

Relation of the behavioural satisfaction $\vDash_{OBS}$ is defined as usual, with the only difference that equality is interpreted in an algebra A as the indistinguishability $\sim_A^{OBS}$ and variables range over the subalgebra A_{OBS}. Formally,

$A \models_{OBS} \phi \iff A/_{\sim_A^{OBS}} \models \phi$. Behavioural semantics is closely related to the observational equivalence of algebras. When only equational specifications are considered, the class of behavioural models coincides with the closure of classical models under observational equivalence (see [3, 5]).

The most straightforward formalization of the concept of implementation refers to the inclusion of model classes: specification $SP' = \langle \Sigma', Ax' \rangle$ *implements* (*refines*) SP, if each model of SP' is a model of SP: $Mod(SP') \subseteq Mod(SP)$ (both SP and SP' are over the same signatures here, $\Sigma' = \Sigma$). This concept has been refined in two ways [12, 13]. First, a notion of *constructor implementation* was proposed: SP' implements SP via a constructor $\kappa : Alg_{\Sigma'} \to Alg_{\Sigma}$, denoted by $SP \overset{\kappa}{\rightsquigarrow} SP'$, if $\bar{\kappa}(Mod(SP')) \subseteq Mod(SP)$ ($\bar{\kappa}(\_)$ is direct image of κ). Intuitively, function κ, called *constructor*, represents a parametrized program, realizing one refinement step. Second, a behavioural realization was taken into account: we say that SP' implements behaviourally SP via κ if

$$\bar{\kappa}(BehMod(SP')) \subseteq BehMod(SP). \tag{1}$$

Appendix A contains an example of such an implementation step. In general, development of a system consists of a sequence of such steps,

$$SP \overset{\kappa_1}{\rightsquigarrow} SP_1 \overset{\kappa_2}{\rightsquigarrow} \dots \overset{\kappa_n}{\rightsquigarrow} SP_n,$$

which finishes when SP_n is the empty specification (e.g. in the implementation of stacks by infinite arrays above, the implementing specification is implicitly assumed to be empty).

2 Regular and Algebraic Sorts

In many practical situations we need only some sorts of a regular algebra to have upper bounds of approximation chains. Assume in the rest of this paper that the set of sorts of Σ is partitioned into two disjoint subsets of *regular* sorts and *algebraic* sorts. The aim is to simplify the work with regular algebras and not to be bothered with considering limit values in algebraic sorts.

In [9] it was argued that the best way to achieve this is to require carrier sets of algebraic sorts to be *essentially flat*. Formally, we say that the carrier set of sort s of a regular Σ-algebra is essentially flat if this algebra is isomorphic (i.e., related via a bijective regular homomorphism) to a regular algebra A whose carrier set of sort s has flat ordering $\leq$ with the least element $\perp_s^A$ (i.e., $a \leq b$ iff $a = \perp_s^A$ or $a = b$). Evidently, isomorphic regular algebras can have different orders.

In the rest of this paper we implicitly assume that some subset of algebraic sorts is distinguished and that all regular algebras considered are essentially flat on those sorts.

3 Finitary Observations

Roughly, finitary observational contexts (i.e., those not containing symbol μ) are powerful enough for indistinguishability in a regular algebra; the only infinitary regular terms really needed are to denote bottoms $\perp$. For fixed OBS, let $\mathcal{F}^{OBS}$ denote the set of all observable contexts $\gamma \in T^{\mu}_{\Sigma}(X \cup \{z_s : s\})$, such that the only possible μ-subterms of γ are of the form $\mu x.x$. In the sequel, by finitary terms we also mean those containing μ-subterms of the form $\mu x.x$.

Theorem 1 ([10]). *In a regular algebra A, the observational indistinguishability $\sim_A^{OBS}$ coincides with the contextual indistinguishability induced in A by contexts from $\mathcal{F}^{OBS}$.*

As a corollary, methods of proving behavioural properties of standard algebras (like context induction [7] or methods developed in [1, 2]) can be reused in the framework of regular algebras.

Let $\Sigma(\perp)$ denote signature Σ enriched by a constant symbol $\perp_s$ in each sort s. For a regular Σ-algebra A, let $|A|_{\Sigma(\perp)}$ denote the standard $\Sigma(\perp)$-algebra with carrier sets and operations as in A and with $\perp_s$ interpreted as $\perp_s^A$ in each sort.

By Theorem 1 we conclude that observational indistinguishability in a regular algebra A coincides with the indistinguishability in $|A|_{\Sigma(\perp)}$. But observational equivalence of regular algebras is not reducible to observational equivalence of standard $\Sigma(\perp)$-algebras: it *does not* hold

$$A \equiv_{OBS} B \quad \Leftrightarrow \quad |A|_{\Sigma(\perp)} \equiv_{OBS} |B|_{\Sigma(\perp)} \tag{2}$$

(note that on the right-hand side $\equiv_{OBS}$ denotes observational equivalence of standard $\Sigma(\perp)$-algebras).

Let us find out where the difficulties are. Since $\sim_A^{OBS}$ is clearly standard $\Sigma(\perp)$-congruence, the forgetful functor $|\_|_{\Sigma(\perp)}$ commutes with observational quotient:

$$|A_{OBS}|_{\Sigma(\perp)}/_{\sim_A^{OBS}} \;=\; |A_{OBS}/_{\sim_A^{OBS}}|_{\Sigma(\perp)}. \tag{3}$$

The observational (standard, regular) subalgebras are generated by (finitary, regular) terms, hence $(|A|_{\Sigma(\perp)})_{OBS} = (|A_{OBS}|_{\Sigma(\perp)})_{OBS}$. Moreover $(|A|_{\Sigma(\perp)})_{OBS}$ may be a *proper* subalgebra of $|A_{OBS}|_{\Sigma(\perp)}$ in general – this is why we need another symbol $\approx_A^{OBS}$ to stand for the indistinguishability (by means of $\Sigma(\perp)$-contexts) in $(|A|_{\Sigma(\perp)})_{OBS}$. However, by Theorem 1 $\approx_A^{OBS}$ and $\sim_A^{OBS}$ agree on $(|A|_{\Sigma(\perp)})_{OBS}$, so the implication from left to right holds in (2). On the other hand, from $|A|_{\Sigma(\perp)} \equiv_{OBS} |B|_{\Sigma(\perp)}$, i.e. from

$$(|A|_{\Sigma(\perp)})_{OBS}/_{\approx^{OBS}} \cong (|B|_{\Sigma(\perp)})_{OBS}/_{\approx^{OBS}}$$

we cannot even conclude (e.g. using (3)) that $|A/_{\sim^{OBS}}|_{\Sigma(\perp)} \cong |B/_{\sim^{OBS}}|_{\Sigma(\perp)}$; but even if we could, this would not guarantee $A/_{\sim^{OBS}} \cong B/_{\sim^{OBS}}$ in general. Intuitively, finitary contexts are more powerful in regular algebras than in standard algebras.

It is common in practical examples that all observable sorts are intended to be essentially flat. Besides practical advantages, essentially flat carriers of observable sorts imply that finitary observational contexts (and consequently standard correspondences) are sufficient for observational equivalence of regular algebras, in contrast to the negative statement (2) above.

Theorem 2 ([10]). *For regular algebras A and B with essentially flat carrier sets of observable sorts, $A \equiv_{OBS} B \quad \Leftrightarrow \quad |A|_{\Sigma(\perp)} \equiv_{OBS} |B|_{\Sigma(\perp)}$.*

As a conclusion, we obtain an effective complete proof technique for observational equivalence of regular algebras:

Corollary 1. *Regular algebras with essentially flat carrier sets of observable sorts are observationally equivalent iff they are related by an observational $\Sigma(\perp)$-correspondence [14].*

The proofs omitted here can be found in [10].

4 Behavioural Implementation of Regular Algebras

In [9] it was argued that when implementation of regular algebras is considered, it is not always possible to reuse the partial order of a model of the implementing specification in a construction of a model of the implemented one. This problem especially arises when an algebraic sort is to implement a regular one, what seems to be common in practical examples. As an example, consider regular algebras as models of the specification of stacks from Section 1 – this opens the possibility to define infinite streams, e.g. $\mu x.push(1, push(0, x))$. Let us look at the following `Pascal`-like implementation of stacks by pointers linking dynamically allocated memory cells.

specification `MEMORY` **extends** `INT`, `BOOL` by
sorts
$\quad memory, address;$
operations
$\quad initmem : memory;$
$\quad null : address;$
$\quad alloc(\_) : memory \to memory \times address;$
$\quad avail(\_,\_) : memory \times address \to bool;$
$\quad \_[\_].val : memory \times address \to int;$
$\quad \_[\_].nxt : memory \times address \to address;$
$\quad (\_[\_].val := \_) : memory \times address \times int \to memory;$
$\quad (\_[\_].nxt := \_) : memory \times address \times address \to memory;$
$\quad \_[\_] \leftarrow \langle \_,\_ \rangle : memory \times address \times int \times address \to memory;$
$\quad copy(\_,\_,\_) : memory \times address \times address \to memory;$
axioms

$$
\begin{array}{l}
a \neq null \Rightarrow avail(initmem, a) = true,\\
alloc(m) = \langle m', a\rangle \;\Rightarrow\; a \neq null \;\wedge\\
\qquad avail(m,a) = true \;\wedge\; avail(m',a) = false \;\wedge\; identical(m,m',a),\\
a \neq null \;\wedge\; m' = (m[a].val := x) \;\Rightarrow\; m'[a].val = x \;\wedge\\
\qquad avail(m',a) = avail(m,a) \;\wedge\; m'[a].nxt = m[a].nxt \;\wedge\; identical(m,m',a),\\
a \neq null \;\wedge\; m' = (m[a].nxt := a'') \;\Rightarrow\; m'[a].nxt = a'' \;\wedge\\
\qquad avail(m',a) = avail(m,a) \;\wedge\; m'[a].val = m[a].val \;\wedge\; identical(m,m',a),\\
m[a] \leftarrow \langle x, a'\rangle = ((m[a].val := x)[a].nxt := a'),\\
copy(m,a,a') = ((m[a'].val := m[a].val)[a'].nxt := m[a].nxt),\\
\\
(\; identical(m,m',a) \iff (\forall a' \in address.\; a \neq a' \Rightarrow\\
\quad avail(m,a') = avail(m',a') \wedge\; m[a'].val = m'[a'].val \;\wedge\; m[a'].nxt = m'[a'].nxt)\;).
\end{array}
$$

For `MEMORY`, a set of observable sorts is $OBS = \{bool, int\}$; all sorts are intended to be essentially flat. We choose a constructor $\kappa_{\texttt{MEMORY}}$ taking a regular algebra over the signature of `MEMORY` to an algebra over the signature of `STACKS`. It is defined by:

$$
\begin{array}{l}
stack := memory \times address,\\
empty := \langle initmem, null\rangle,\\
top(\langle m,a\rangle) := \ \textbf{if}\ \ a = null \vee a = \bot_{address}\ \ \textbf{then}\ \ \bot_{int}\ \textbf{else}\ \ m[a].val,\\
pop(\langle m,a\rangle) := \ \textbf{if}\ \ a = null \vee a = \bot_{address}\ \ \textbf{then}\ \ \langle m,a\rangle\ \textbf{else}\ \ \langle m, m[a].nxt\rangle,\\
push(x,\langle m,a\rangle) := \ \textbf{let}\ \ \langle m',a'\rangle = alloc(m)\ \ \textbf{in}\ \ \langle m'[a'] \leftarrow \langle x,a\rangle, a'\rangle.
\end{array}
$$

It implements in a straightforward manner all finite stacks, but what about infinite ones? Are they realizable in (behavioural) models of MEMORY? In this particular case a right idea is to exploit cyclic lists; however, a more universal method is needed in general.

For a given model of MEMORY, consider an algebra A yielded by $\kappa_{\mathtt{MEMORY}}$. Roughly, our idea is to define a semantical counterpart of μ-operator, that is for each function $f : |A|_{stack} \to |A|_{stack}$, a value $\mathsf{fix}_{stack}(f)$ in $|A|_{stack}$. This is enough to define values of all regular terms, since we can take $\mathsf{fix}_{stack}(f)$ as the value of a μ-term $\mu x.t$ under a valuation v, for an appropriate function f induced by t and v. Formally, we can define inductively the value of $(\mu x.t)_{A[v]}$ by: $(\mu x.t)_{A[v]} := \mathsf{fix}_{stack}(\lambda a.t_{A[v_a]})$, where v_a extends v by $v_a(x) = a$. In our example this could look like

$$
\begin{array}{l}
\mathsf{fix}_{stack}(f) := \ \textbf{let}\ \ \langle m',a'\rangle = alloc(initmem),\ \ \langle m,a\rangle = f(\langle m',a'\rangle)\ \ \textbf{in}\\
\qquad\qquad \textbf{if}\ \ a \neq a'\ \ \textbf{then}\ \ \langle copy(m,a,a'), a'\rangle\ \textbf{else}\ \ \langle initmem, \bot_{address}\rangle.
\end{array}
$$

This method of defining values of all μ-terms will be called *μ-induction* in the sequel. In general, it is sufficient to define $\bot$ in each algebraic sort and a fix operator for each regular sort. For regular sorts s, we derive $\bot_s$ for instance by $\bot_s = \mathsf{fix}_s(\mathrm{id}_{A_s})$ (e.g. $\bot_{stack} = \langle initmem, \bot_{address}\rangle$); for algebraic sorts s, one can assume that $\mathsf{fix}_s(f) = f(\bot_s)$. Hence μ-induction in an algebra A needs a family $\{\bot_s\}$ indexed by algebraic sorts s and a family of functions, $\mathsf{fix}_s : {|A|_s}^{|A|_s} \to |A|_s$, indexed by regular sorts s.

According to the terminology used below, A defined in this way is a *pre-regular algebra*, differing from a regular algebra in at least one respect: its carrier sets are not ordered and consequently it does not satisfy continuity; in particular, values of μ-terms given by μ-induction do not have to be fixed points. In what follows, we introduce pre-regular algebras and use them in behavioural implementation step of regular algebras.

4.1 Pre-Regular Algebras

Let S denote the set of sorts of a fixed signature Σ. It is reasonable to consider an abstract syntax given by the set $\mathfrak{T}^{\mu}_{\Sigma}(X)$ of regular terms up to α-conversion, equal to the quotient of $T^{\mu}_{\Sigma}(X)$ by all equalities of the form: $\mu x.t(x) = \mu y.t(y)$, for all terms t and variables x, y. The mapping $X \mapsto \mathfrak{T}^{\mu}_{\Sigma}(X)$ can be extended to an endofunctor $\mathfrak{T}^{\mu}_{\Sigma}$ in Set^S, similarly to an endofunctor $X \mapsto T_{\Sigma}(X)$; moreover, in a similar way, $\mathfrak{T}^{\mu}_{\Sigma}$ can be extended to a monad, i.e., equipped with unit and multiplication. By μ-induction we define in fact an algebra of this monad - $\mathfrak{T}^{\mu}_{\Sigma}$-algebras are called *pre-regular algebras* in the sequel. By the very definition, a pre-regular algebra has enough structure to assign values to all regular terms, in a canonical way. Morphisms of $\mathfrak{T}^{\mu}_{\Sigma}$-algebras are precisely those functions which preserve values of all regular terms, similarly as regular homomorphisms: h is a morphism from A to B iff $h(t_{A[v]}) = t_{B[h \circ v]}$, for any regular term t and valuation v in A (this category is denoted $\mathit{Pre\text{-}RAlg}_{\Sigma}$ in the sequel).

Each regular algebra is (via the evident forgetful functor) a pre-regular algebra; in fact, category RAlg_{Σ} is equivalent to a full subcategory of pre-regular algebras. Moreover, the forgetful functor has a left adjoint:

Proposition 1. *RAlg_{Σ} is a reflective subcategory of $\mathit{Pre\text{-}RAlg}_{\Sigma}$.*

Proof. Observe that the notion of pre-congruence from Section 1 is meaningful in a pre-regular algebra. By the quotient of a pre-regular algebra A by a pre-congruence $\sqsubseteq$, denoted by $A/\sqsubseteq$, we mean the quotient by a congruence $\sqsubseteq \cap \sqsubseteq^{-1}$ induced by $\sqsubseteq$; $A/\sqsubseteq$ is a regular algebra, i.e., can be appropriately equipped with a partial order, similarly as in Section 1.

Exploiting these facts, the left adjoint to the forgetful functor is given by the quotient of a pre-regular algebra A by the smallest pre-congruence $\preccurlyeq_A$ in A, $A \mapsto A/\preccurlyeq_A$. This mapping is functorial. To see this, notice that every morphism $f : A \to B$ of pre-regular algebras together with a pre-congruence $\sqsubseteq$ on B induces a pre-congruence: $\{(a, a') : f(a) \sqsubseteq f(a')\}$ on A.

Since $\preccurlyeq_A$ is the smallest pre-congruence, there exists a unique morphism $A/_{\preccurlyeq_A} \to B/_{\preccurlyeq_B}$ making the square on the right commute (horizontal maps are quotient projections). Moreover, by the very definition, this induced morphism is a regular homomorphism.

$$\begin{tikzcd} A \arrow[r, "\pi_A"] \arrow[d, "f"'] & A/_{\preccurlyeq_A} \arrow[d, "!"] \\ B \arrow[r, "\pi_B"] & B/_{\preccurlyeq_B} \end{tikzcd}$$

When B is regular, $\preccurlyeq_B$ induces identity congruence; thus, by instantiating the diagram above we immediately get a 1-1 correspondence between homsets $\mathit{Hom}_{\mathit{Pre\text{-}RAlg}_{\Sigma}}(A, B)$ and $\mathit{Hom}_{\mathit{RAlg}_{\Sigma}}(A/\preccurlyeq_A, B)$. Hence the quotient functor $\_/\preccurlyeq_{\_}$ is the left adjoint to the embedding of RAlg_{Σ} into $\mathit{Pre\text{-}RAlg}_{\Sigma}$; the projections π form a unit. $\square$

Notions of subalgebra, quotient, etc. can be straightforwardly extended to pre-regular algebras. The relation $\models_{OBS}$ of behavioural satisfaction can be lifted to pre-regular algebras in a natural way. By $A \models_{OBS} \phi$, for pre-regular A, we mean that $A/_{\sim_A^{OBS}} \models \phi$, where $\sim_A^{OBS}$ denotes the indistinguishability induced by the set of *all* observable contexts (including μ-terms) in the pre-regular subalgebra A_{OBS} of A generated by OBS. (A_{OBS} contains precisely those elements that are a value of some regular term with only observable variables.) Note that it is indispensable to take all contexts into account, since values of μ-terms in a pre-regular algebra can be defined arbitrarily and we can not use continuity, as in the case of regular algebras in Theorem 1, to restrict the set of relevant contexts. Moreover, considering all regular contexts guarantees that $\sim_A^{OBS}$ is a congruence on A_{OBS} (in fact, the greatest congruence which is identity on OBS). In the following section we overload symbol $\equiv_{OBS}$ to denote also the equivalence of pre-regular algebras factorized by indistinguishabilities $\sim_A^{OBS}$ similarly as stated for regular algebras in Definition 1.

5 Constructor Implementation Step

By now, we found pre-regular algebras useful in the behavioural implementation of regular algebras. Motivated by examples and by Proposition 1, we propose an implementation methodology for regular algebras, as a slight adaptation of the behavioural constructor implementation of standard algebras presented in Section 1. By the abuse of notation, given a specification SP over Σ, by $BehMod(SP)$ we mean the class of all regular Σ-algebras which satisfy behaviourally SP and whose algebraic sorts are essentially flat.

Let SP, SP' be two specifications over Σ and Σ', respectively. Let OBS denote the subset of observable sorts in SP. In order to express the way how SP' implements (refines) SP, we define a constructor $\kappa : RAlg_{\Sigma'} \to Pre\text{-}RAlg_{\Sigma}$, for instance by μ-induction. This mapping induces a function $\kappa' : RAlg_{\Sigma'} \to RAlg_{\Sigma} : A \mapsto \kappa(A)/\preccurlyeq_{\kappa(A)}$. But for some technical reason, which becomes apparent in (8) below, instead of $\preccurlyeq_{\kappa(A)}$ we prefer to quotient $\kappa(A)$ by the smallest pre-congruence on the observable subobject, that is on the subobject $\kappa(A)_{OBS}$ of $\kappa(A)$ generated by the observable sorts[2]:

$$\kappa'(A) := \kappa(A)_{OBS}/\preccurlyeq_{\kappa(A)_{OBS}}.$$

Then, we say that SP' implements behaviourally SP via κ if

$$\bar{\kappa'}(BehMod(SP')) \subseteq BehMod(SP). \tag{4}$$

5.1 Behavioural Consistency

Similarly as in the case of standard algebras, for the correctness of the constructor implementation step one needs to show (4). When $SP = \langle \Sigma, Ax \rangle$ is a basic specification given by a set of axioms, (4) can be proved for instance by showing that $\kappa'(A)$ satisfies behaviourally axioms of SP, for any behavioural model A of SP'. However, this is difficult in practice, since the congruence in the construction of $\kappa'(A)$ is not given explicitly – the task would be simpler if we could consider $\kappa(A)$ instead of $\kappa'(A)$. This would be the case when $\kappa(A)$ and $\kappa'(A)$ satisfy behaviourally the same formulas:

$$\{\phi : \kappa(A) \models_{OBS} \phi\} = \{\phi : \kappa'(A) \models_{OBS} \phi\}. \tag{5}$$

We say that constructor κ is *behaviourally consistent* if (5) holds for all A in $BehMod(SP')$. A sufficient condition for (5) is:

$$\kappa(A) \equiv_{OBS} \kappa'(A), \quad \text{for each } A \in BehMod(SP'), \tag{6}$$

[2] $\preccurlyeq_{\kappa(A)_{OBS}}$ needs not coincide with $\preccurlyeq_{\kappa(A)}$ restricted to $\kappa(A)_{OBS}$ – the latter is coarser in general.

where $\equiv_{OBS}$ denotes the equivalence of pre-regular algebras factorized by the indistinguishabilities $\sim_A^{OBS}$. The rest of this section is devoted mainly to formulating of a condition sufficient for (6).

In the sequel let $B := \kappa(A)_{OBS}$ denote the observable subalgebra of the pre-regular algebra yielded by κ, for an arbitrary fixed $A \in BehMod(SP')$. Since $\kappa(A) \equiv_{OBS} B$, (6) is equivalent to

$$B \equiv_{OBS} B/\preccurlyeq_B. \tag{7}$$

Hence, for behavioural consistency (in fact for (7)) it is sufficient to know that $\preccurlyeq_B$ induces an observational congruence on B:

$$\preccurlyeq_B \cap \preccurlyeq_B^{-1} \subseteq \sim_B^{OBS}. \tag{8}$$

(8) states that the congruence induced by $\preccurlyeq_B$ is identity on observable sorts; in other words, imposing continuity on B does not lead to identification of observable elements. From the practical perspective this is a natural condition, since it conforms neatly to the situation when observable sorts are to support some standard datatypes, which rest unaffected during a refinement step.

For (8) to hold, $\sim_B^{OBS}$ should be necessarily a total congruence (i.e., $B = B_{OBS}$). Fortunately, it is so since B is generated by OBS (recall definition of κ' in the beginning of this section).

Evidently, $\sim_B^{OBS}$ is equal to $\lesssim_B^{OBS} \cap (\lesssim_B^{OBS})^{-1}$, where the *observational preorder* $\lesssim_B^{OBS}$ is defined analogously to $\sim_B^{OBS}$: for $b, b' \in |B|_s$, $b \lesssim_B^{OBS} b'$ if and only if for all valuations $v : X \to |B|_{OBS}$ and all regular observable contexts γ, $\gamma_{B[v]}(b) \leq \gamma_{B[v]}(b')$; $\leq$ stands here for the flat order in all observable sorts.[3] Having this, we conclude that for (8) it is sufficient to have $\preccurlyeq_B \subseteq \lesssim_B^{OBS}$. Now, recalling that $\preccurlyeq_B$ is the smallest pre-congruence on B, we deduce:

Lemma 1. *A constructor κ is behaviourally consistent whenever $\lesssim_{\kappa(A)}^{OBS}$ is a pre-congruence on $\kappa(A)_{OBS}$, for each $A \in BehMod(SP')$.*

From now on we develop a sufficient condition for this to hold. Since B is generated by OBS, we can suitably represent each non-observable value by a term with only observable variables. This implies that operations in B are monotonic w.r.t. $\lesssim_B^{OBS}$, hence $\sim_B^{OBS}$ is preserved by operations. We need to check the other condition from the definition of pre-congruence (cf. Section 1). For any recursor $q \in |T_\Sigma^\mu(X \cup \{z : s\})|_s$ and valuation $v : X \to |B|$, let $b_i := q^i_{B[v]}(\bot)$, for $i = 0, 1, \ldots$, $b := (\mu z.q)_{B[v]}$; then we should show that

> for each $t \in |T_\Sigma(X \cup \{y : s\})|_{s'}$ and $u : X \to |B|$, $\{t_{B[u_i]}\}_{i \in \omega}$ is a chain w.r.t. $\lesssim_B^{OBS}$ with a l.u.b. $t_{B[u']}$ (u_i, $i \in \omega$, and u' extend u by $u_i(y) = b_i$ and $u'(y) = b$).

As B is generated by OBS, we may replace each occurrence of a non-observable variable in t (besides y) by a term with only observable variables; hence the

[3] Surprisingly, at this point we could use *any* partial order instead of $\leq$.

mapping $a \mapsto t_{B[u[y \mapsto a]]}$ preserves $\lesssim_B^{OBS}$. This allows us to replace condition above with a simpler one:

$$\{b_i\}_{i \in \omega} \text{ is a chain w.r.t. } \lesssim_B^{OBS} \text{ with a least upper bound } b.$$

Moreover, the mapping $a \mapsto q_{B[v[z \mapsto a]]}$ preserves $\lesssim_B^{OBS}$ by the same argument as above, hence for the last formula it suffices that the following two conditions hold:

$$\bot_s^B \lesssim_B^{OBS} b', \quad \text{for all } b' \in |B|_s, \tag{9}$$

$$b \;=\; \bigsqcup_{i \in \omega} b_i \quad \text{w.r.t. } \lesssim_B^{OBS}. \tag{10}$$

The observable behaviour of an element of B consists of the infinite tuple of values yielded by all observable contexts (with all valuations into observable sorts) applied to it. As $\lesssim_B^{OBS}$ is the point-wise pre-order, (9) and (10) say that a bottom $\bot$ has the smallest observable behaviour and that the observable behaviour of b is the least upper bound of behaviours of b_i. This allows us to formulate a methodological paradigm:

> *A constructor is guaranteed to be behaviourally consistent whenever $\bot$ in each sort has the smallest observable behaviour and the observable behaviour of each (candidate for) least upper bound of an approximation chain is the least upper bound of the observable behaviours of approximants.*

Expanding definition of $\lesssim_B^{OBS}$ and recalling that $\leq$ was chosen to be flat, the paradigm can be formalized as follows:

Theorem 3. *A constructor κ is behaviourally consistent if for each behavioural model A of SP' it holds: for all contexts $\gamma \in T_\Sigma^\mu(X \cup \{z_s : s\})_o$ of an observable sort o and valuations w into observable carriers of $B := \kappa(A)_{OBS}$,*

- $\gamma_{B[w]}(\bot_s) \neq \bot_o \Rightarrow \forall b \in B_s.\ \gamma_{B[w]}(b) = \gamma_{B[w]}(\bot_s)$,
- *for any $q \in T_\Sigma^\mu(X \cup \{z : s\})_s$, $v : X \to |B|$, let $b_i := q^i_{B[v]}(\bot)$, $i \in \omega$ and $b := (\mu z.q)_{B[v]}$; then*
$$\forall i \in \omega.(\gamma_{B[w]}(b_i) \neq \bot_o \Rightarrow \gamma_{B[w]}(b) = \gamma_{B[w]}(b_i)),$$
$$(\forall i \in \omega.\gamma_{B[w]}(b_i) = \bot_o) \Rightarrow \gamma_{B[w]}(b) = \bot_o.$$

For instance, $\kappa_{\texttt{MEMORY}}$ from the beginning of previous section is behaviourally consistent.

5.2 Finitary Contexts

Theorem 3 concerns all regular observable contexts γ. From practical perspective, especially when considering proof methods for behavioural properties, it would be useful to be able to restrict to only finitary contexts. Assume for a while that we replaced $\lesssim_B^{OBS}$ in Lemma 1 with pre-order induced by only contexts from $\mathcal{F}^{OBS}$ (similarly as in Theorem 1 in Section 3). If we prove now that this pre-order is a pre-congruence, then the induced congruence, say $\sim$, necessarily coincides with $\sim_B^{OBS}$. This follows from the fact that $\sim$ is obviously coarser than $\sim_B^{OBS}$, $\sim_B^{OBS} \subseteq \sim$, but on the other hand it is necessarily identity on OBS. Since $\sim_B^{OBS}$ is the greatest congruence being identity on OBS, we get $\sim = \sim_B^{OBS}$ (this idea of identifying a sufficient subset of "crucial" contexts comes e.g. from [1]). From this we conclude that Lemma 1 would still hold and behavioural consistency would follow.

Unfortunately, our further considerations (culminating in Theorem 3) are no longer valid when only contexts from $\mathcal{F}^{OBS}$ are taken into account. Infinitary regular terms are indispensable when valuation maps a variable to value in B not representable by a finitary term with observable variables. Consequently, we can only restrict the proof obligation stated in Theorem 3 to those contexts γ that contain no occurrence of the context variable inside a μ-term.

Surprisingly, finitary contexts are sufficient for proving correctness of implementation when we already know that κ is behaviourally consistent. Namely, by (5), every proof in $\kappa(A)$ involving observational contexts can be carried over to $\kappa'(A)$, where finitary contexts suffice by Theorem 1.

6 Final Remarks

The main result of this paper is an adaptation of the behavioural constructor implementation methodology to the framework of regular algebras. A proof obligation was given that guarantees (together with results on finitary character of observational indistinguishability) that the proof of correctness of an implementation step is feasible.

The subject needs still more studies, especially the issue of proving correctness of implementation. In particular, we did not tackle the task of proving that carriers of algebraic sorts are essentially flat in the implementation step. Moreover, an interesting topic for further research is to investigate relationship between regular algebras and coalgebraic specifications [11] and to try to apply proof methods used there. It could be probably of some relevance here that the initial regular Σ-algebra can be seen as a suitable sub-coalgebra of the final $\Sigma(\perp)$-coalgebra, hence it admits the coinduction principle.

References

1. Bidoit, M., Hennicker, R. Behavioural theories and the proof of behavioural properties. *Theoretical Computer Science* 165(1): 3-55, 1996.
2. Bidoit, M., Hennicker, R. Modular correctness proofs of behavioural implementations. *Acta Informatica* 35(11):951-1005, 1998.
3. Bidoit, M., Hennicker, R., Wirsing, M. Behavioural and abstractor specifications. *Science of Computer Programming* 25(2-3), 1995.
4. Bidoit, M., Tarlecki, A. Regular algebras: a framework for observational specifications with recursive definitions. Report LIENS-95-12, Ecole Normale Superieure, 1995.

5. Bidoit, M., Tarlecki A. Behavioural satisfaction and equivalence in concrete model categories. *Manuscript.* A short version appeared in *Proc. 20th Coll. on Trees in Algebra and Computing* CAAP'96, Linköping, LNCS 1059, 241-256, Springer-Verlag, 1996.
6. Guessarian, I., Parisi-Presicce, F. Iterative vs. regular factor algebras. *SIGACT News* 15(2), 32-44, 1983.
7. Hennicker, R. Context induction: A proof principle for behavioural abstractions and algebraic implementations. *Formal Aspects of Computing*, 3(4):326-345, 1991.
8. Knapik, T. Specifications algebriques observationnelles modulaires: une semantique fondee sur une relation de satisfaction observationnelle. PhD Thesis, Ecole Normal Superieur, Paris, 1993.
9. Lasota, S. *Algebraic observational equivalence and open-maps bisimilarity.* PhD Thesis, Institute of Informatics, Warsaw University, March 2000. Accessible at `http://www.mimuw.edu.pl/~sl/work/phd.ps.gz`.
10. Lasota, S. Finitary observations in regular algebras. To appear in Proc. 27th Seminar on Current Trends in Theory and Practice of Informatics *SOFSEM*'2000, LNCS. Accessible at `http://www.mimuw.edu.pl/~sl/papers`.
11. Rutten, J.J.M.M. Universal coalgebra: a theory of systems. CWI Report CS-R9652, 1996.
12. Sannella, D., Tarlecki, A. Towards formal development of programs from algebraic specifications: implementations revisited. *Acta Informatica* 25:233-281, 1988.
13. Sannella, D., Tarlecki, A. Essential Concepts of Algebraic Specification and Program Development. *Formal Aspects of Computing* 9:229-269, 1997.
14. Schoett, O. *Data abstraction and correctness of modular programming.* PhD thesis, CST-42-87, Department of Computer Science, University of Edinburgh, 1987.
15. Tarlecki, A., Wirsing, M. Continuous abstract data types. *Fundamenta Informaticae* 9(1986), 95-126.
16. Tiuryn, J. Fixed-points and algebras with infinitely long expressions. Part I. Regular algebras. *Fundamenta Informaticae* 2(1978), 102-128.
17. Tiuryn, J. Fixed-points and algebras with infinitely long expressions. Part II. Muclones of regular algebras. *Fundamenta Informaticae* 2(1979), 317-336.

A An Example of Behavioural Constructor Implementation

As an example, consider a specification of queues and its behavioural implementation by pairs of stacks.

specification QUEUES **extends** INT, BOOL by
sorts
$queue$;
operations
$empty\_queue : queue$;
$empty\_queue?(\_) : queue \to bool$;
$put(\_,\_) : queue \times int \to queue$;
$get(\_) : queue \to int$;
$rest(\_) : queue \to queue$;
axioms
$empty\_queue?(empty\_queue) = true$
$empty\_queue?(put(q, e)) = false$
$empty\_queue?(q) \Rightarrow get(put(q, e)) = e \ \wedge \ rest(put(q, e)) = q$
$\neg empty\_queue?(q) \Rightarrow get(put(q, e)) = get(q) \ \wedge \ rest(put(q, e)) = put(rest(q), e)$

A constructor behavioural implementation of `QUEUES` by `STACKS` may be given by the following definitions (some ad hoc, but hopefully self-explanatory notation is used here to define a function that maps Sig(`STACKS`)-algebras to Sig(`QUEUES`)-algebras):

$$
\begin{array}{l}
queue := stack \times stack \\
empty\_queue := \langle empty, empty\rangle \\
empty\_queue?(\langle s_1, s_2\rangle) := empty?(s_1) \ \wedge\ empty?(s_2) \\
put(\langle s_1, s_2\rangle, e) := \langle push(e, s_1), s_2\rangle \\
get(\langle s_1, s_2\rangle) := \\
\qquad\quad \textbf{if}\ \ empty?(s_2)\ \textbf{then}\quad \textbf{let}\ \ s_2' := reverse(\langle s_1, s_2\rangle)\ \ \textbf{in}\ \ top(s_2') \\
\qquad\quad \textbf{else}\ \ top(s_2) \\
rest(\langle s_1, s_2\rangle) := \\
\qquad\quad \textbf{if}\ \ empty?(s_2)\ \textbf{then}\quad \textbf{let}\ \ s_2' := reverse(\langle s_1, s_2\rangle)\ \ \textbf{in}\ \ \langle empty, pop(s_2')\rangle \\
\qquad\quad \textbf{else}\ \ \langle s_1, pop(s_2)\rangle
\end{array}
$$

$$
\begin{array}{l}
reverse(\langle empty, s\rangle) = s \\
reverse(\langle push(n, s_1), s_2\rangle) = reverse(\langle s_1, push(n, s_2)\rangle)
\end{array}
$$

This is a correct implementation only when behavioural satisfaction is assumed, w.r.t. observable sorts $\{int, bool\}$.

If we intend specifications `STACKS` and `QUEUES` to describe regular algebras, we ought to extend the implementation step by a definition of fix_{queue}. An infinite queue behaves like an infinite stack, with $get(\_)$ and $rest(\_)$ operations corresponding to $pop(\_)$ and $top(\_)$, respectively. Moreover, putting new elements into an infinite queue has no effect. Hence we may essentially re-use the fix operator of stacks:

$$\mathsf{fix}_{queue}(f) := \langle\ \ empty\ \ ,\ \ \mathsf{fix}_{stack}(\ \lambda s.reverse(\ f(\langle empty, s\rangle)\ )\ )\ \ \rangle.$$

In particular, we derive $\perp_{queue} = \langle empty, \perp_{stack}\rangle$.

Some larger examples of specification and implementation in the setting of regular algebras can be found in [9].