Short Contributions (CWI Technical report SEN-1004)

10th International Workshop on

Coalgebraic Methods in Computer Science

Paphos, Cyprus

March 26–28, 2010

Editors:

B.P.F. Jacobs
J.J.M.M Rutten
M. Niqui
A. Silva

Contents

Algebraic Enriched Coalgebras

Filippo Bonchi * Marcello Bonsangue † Jan Rutten ‡ Alexandra Silva§

Abstract

The powerset construction is a standard method for converting a non-deterministic automaton into an equivalent deterministic one as far as language is concerned. In this short paper, we lift the powerset construction on automata to the more general framework of coalgebras with enriched state spaces. Examples of applications includes transducers, and Rabin probabilistic automata.

1 Introduction

Coalgebra is by now a well established general framework for the study of the behaviour of large classes of dynamical systems. Coalgebras generally come equipped with a standard notion of (behavioural) observational equivalence ($\sim_F$) that is fully determined by their (functor) type F. Moreover, for many functors F there exists a *final* coalgebra into which any F-coalgebra is mapped by a unique homomorphism that identifies all F-bisimilar states.

For example, deterministic automata (DA), are coalgebras of the functor $DA(X) = 2 \times X^A$. In a DA, two states are DA-bisimilar precisely when they accept the same language. The set 2^{A^*} of all formal languages constitutes a final DA-coalgebra, into which every DA is mapped by a homomorphism that sends any state to the language it accepts.

It is well-known that *non-deterministic* automata (NDA) often provide more efficient (smaller) representations of formal languages than DA's. Language acceptance of NDA's is typically defined by turning them into DA's via the *powerset construction*. Coalgebraically this works as follows. NDA's are coalgebras of the functor $NDA(X) = 2 \times \mathscr{P}_\omega(X)^A$, where $\mathscr{P}_\omega$ is the finite powerset. An NDA-coalgebra $(X, f\colon X \to 2 \times \mathscr{P}_\omega(X)^A)$ is *determinized* by transforming it into a DA-coalgebra $(\mathscr{P}_\omega(X), f^\sharp\colon X \to 2 \times \mathscr{P}_\omega(X)^A)$. Here $f^\sharp$ is the unique semilattice morphism such that $f^\sharp(\{s\}) = f(s)$ for every state s of X. Then, the language accepted by a state s i n the NDA (X, f) is defined as the language accepted by the state $\{s\}$ in the DA $(\mathscr{P}_\omega(X), f^\sharp)$.

In this short paper we lift the powerset construction on automata to the more general framework of coalgebras with state spaces enriched over an algebraic structure. Examples of application include transducers, and Rabin probabilistic automata.

2 Algebraic enriched coalgebras

We consider coalgebras $f\colon X \to FT(X)$ for a functor F and a monad $\mathbf{T}$ such that F has a lifting F^* to the Eilemberg-Moore category of $\mathbf{T}$-algebra (or, equivalently, there is a distributive law $\lambda : TF \Rightarrow FT$ [4]). This implies that $FT(X)$ is the carrier of a $\mathbf{T}$-algebra. The inter-play between the transition type F and the computational type $\mathbf{T}$ will allow each coalgebra $f\colon X \to FT(X)$ to be extended uniquely to a T-algebra morphism $f^\sharp\colon (T(X), \mu_X) \to (FT(X), h)$ such that $f^\sharp \circ \eta_X = f$, where η_X is the unit of the monad $\mathbf{T}$. Intuitively, $\eta_X : X \to T(X)$ is the inclusion of the state space of the coalgebra $f\colon X \to FT(X)$ into the enriched state space $T(X)$, and $f^\sharp\colon T(X) \to FT(X)$ is the extension of the coalgebra f to $T(X)$.

*INRIA Saclay - LIX, École Polytechnique
†LIACS - Leiden University
‡Centrum Wiskunde en Informatica (CWI)

We say that two elements x_1 and x_2 in X are *F-equivalent with respect to a monad* $\mathbf{T}$, written $x_1 \approx_F^T x_2$, if and only if $\eta_X(x_1) \sim_F \eta_X(x_2)$. The equivalence $\sim_F$ is just observational equivalence for the F-coalgebra $f^\sharp\colon T(X) \to FT(X)$.

If the functor F has a final coalgebra (Ω, ω) , we can capture the semantic equivalence above in the following commuting diagram

$$\begin{array}{ccccc} X & \xrightarrow{\eta_X} & T(X) & \overset{[\![-]\!]}{\dashrightarrow} & \Omega \\ {\scriptstyle f}\downarrow & \swarrow_{f^\sharp} & & & \downarrow{\scriptstyle \omega} \\ FT(X) & & \dashrightarrow & & F(\Omega) \end{array} \tag{1}$$

Back to our example in the introduction, two states x_1 and x_2 of a NDA (in which T is instantiated to $\mathscr{P}_\omega$ and F to $2 \times Id^A$) would satisfy $x_1 \approx_F^T x_2$ if and only if they recognize the same language).

If Ω can be constructed as the limit of the final sequence, then Ω is the final F-coalgebra if and only if Ω is the carrier of the final coalgebra of the lifted functors F^* over $\mathbf{T}$-algebras. Further, the unique F-coalgebra homomorphism $[\![-]\!]$ as in diagram (1) is a T-algebra homomorphism. Thus, F^*-observational equivalence coincides with F-equivalence with respect to $\mathbf{T}$. The latter equivalence is stronger than the FT-equivalence. For NDA this instantiates to the well-known fact that bisimulation implies language equivalence. Note that, in general, the above inclusion is strict.

Our framework is substantially different from the coalgebraic trace approach presented in [3]. There, instead of considering FT-coalgebras and $\mathbf{T}$-algebras, TF-coalgebras and free $\mathbf{T}$-algebras are studied. Only under some additional assumptions, the coincidence of the two semantics is ensured.

Example: Transducers Consider the monad $T(X) = \mu v.X + (O \times v) \cong O^* \times X$, with O a set of outputs, unit $\eta(x) = \langle \varepsilon, x \rangle$ and, for $f\colon X \to T(Y)$, $f^\sharp\colon T(X) \to T(Y)$ is given by $f^\sharp(\langle w, x \rangle) = \langle ww', x' \rangle$ where $\langle w', x' \rangle = f(x)$. For $F = O^* \times id^A$, a (total) subsequential transducer [2] is an FT-coalgebra $\langle o, t \rangle\colon X \to O^* \times (O^* \times X)^A$: $o\colon X \to O^*$ is the terminal output function; $t\colon X \to (O^* \times X)^A$ is the pairing of the output function and the next state-function.

The behaviour of a state x will be given by $[\![\eta(x)]\!] = [\![\langle \varepsilon, x \rangle]\!]$, where, for every $\langle w, x \rangle \in O^* \times X$, $[\![\langle w, x \rangle]\!]\colon A^* \to O^*$, is given by

$$\begin{array}{lcl} [\![\langle w, x \rangle]\!](\varepsilon) & = & w \cdot o(x) \\ [\![\langle w, x \rangle]\!](aw_1) & = & w \cdot ([\![t(x)(a)]\!](w_1)) \end{array}$$

Example: Rabin probabilistic automata. Consider the discrete probability distribution monad $T(X) = \mathscr{D}_\omega(X)$, with unit η the Dirac distribution and, for $f\colon X \to T(Y)$, $f^\sharp\colon T(X) \to T(Y)$ is given by

$$f^\sharp(c) = \lambda y. \sum_{d \in \mathscr{D}_\omega(Y)} \left(\sum_{x \in f^{-1}(d)} c(x) \right) \times d(y)$$

For $F = [0,1] \times id^A$, a Rabin probabilistic automaton [6] is a coalgebra $\langle o, t \rangle\colon X \to [0,1] \times \mathscr{D}_\omega(X)^A$ (note that $\mathscr{D}_\omega(2) \cong [0,1]$): each state x has an output value in $o(x) \in [0,1]$ and, for each input a, $t(x)(a)$ is a probability distribution of next states. The behaviour of a state x is given by $[\![\eta(x)]\!]\colon A^* \to [0,1]$, defined below. Intuitively, one can think of $[\![\eta(x)]\!]$ as a probabilistic language: each word is associated with a value $p \in [0,1]$.

$$\begin{array}{lcl} [\![d]\!](\varepsilon) & = & \sum_{b \in [0,1]} \left(\sum_{o(x)=b} d(x) \right) \times b \\ [\![d]\!](aw) & = & [\![\lambda x'. \sum_{c \in \mathscr{D}_\omega(X)} (\sum_{b=t(x)(a)} d(x)) \times c(x'))]\!](w) \end{array}$$

It is worth to note that this exactly captures the semantics of [6], while the ordinary $\sim_{FT}$ coincides with *probabilistic bisimilarity* of [5].

References

[1] J. Adámek. Free algebras and automata realization in the language of categories. *Comment. Math. Univ. Carolinae*, 15:589–602, 1974.

[2] H. H. Hansen. Coalgebraising subsequential transducers. *Electr. Notes Theor. Comput. Sci.*, 203(5):109–129, 2008.

[3] I. Hasuo, B. Jacobs, and A. Sokolova. Generic trace semantics via coinduction. *Logical Methods in Computer Science*, 3(4), 2007.

[4] P.T. Johnstone. Adjoint lifting theorems for categories of algebras. *Bulletin London Mathematical Society*, 7:294–297, 1975.

[5] K. G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[6] M. O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

Similarity quotients as final coalgebras (work in progress)

Paul Blain Levy, University of Birmingham pbl@cs.bham.ac.uk

We first give, without proof, some results about bisimulation, simulation and 2-nested simulation, then consider the general theory in which they could be formulated and proved.

Let Act be a countable set, and let F be the endofunctor on **Set** mapping X to $\mathscr{P}^{\leqslant \aleph_0}(\mathsf{Act} \times X)$. Thus an F-coalgebra is a countably branching Act-labelled transition system.

1 Bisimulation (standard results)

Given F-coalgebras A, B, we write $\backsimeq_{A,B}$ for the largest bisimulation from A to B.

Proposition 1 (Using a final F-coalgebra)**.** *Fix a final F-coalgebra $\nu F = (P, \zeta)$, and for any F-coalgebra A we write $A \xrightarrow{c_A} \nu F$ for the unique anamorphism.*

Let $A = (X, \theta)$ and $B = (Y, \phi)$ be F-coalgebras, and let $x \in X$ and $y \in Y$. Then $x \backsimeq_{A,B} y$ iff $c_A x = c_B y$.

Proposition 2 (Constructing a final F-coalgebra)**.** *Suppose that $A = (X, \theta)$ is "big enough" up to bisimilarity, i.e. for every F-coalgebra $B = (Y, \phi)$ and $y \in Y$ there is $x \in X$ such that $x \backsimeq_{A,B} y$.*

Write Y for the quotient of X by $\backsimeq_{A,A}$, and write $X \xrightarrow{p} Y$ for the canonical map. Then there is a unique map $Y \xrightarrow{\phi} FY$ making $A \xrightarrow{p} (Y, \phi)$ a coalgebra morphism. Moreover (Y, ϕ) is final.

For example, let A be the coproduct of all final coalgebras carried by an initial segment of $\mathbb{N}$. This is big enough up to bisimilarity because any node of an F-coalgebra has only countably many descendants.

2 Simulation

Given F-coalgebras A and B, we write $\lesssim_{A,B}$ for the largest simulation from A to B.

Write **Preord** for the category of preordered sets and monotone maps, and **Poset** for the full subcategory of posets. We write $Q : \mathbf{Preord} \longrightarrow \mathbf{Poset}$ for the reflection.

For any relation $X \xrightarrow{\mathscr{R}} Y$, we define a relation $FX \xrightarrow{\mathscr{P}^{\mathsf{sim}}} FY$ to be

$$\{(u, v) \mid \forall (a, x) \in u.\ \exists y \in Y.\ (a, y) \in v \wedge (x, y) \in \mathscr{R}\}$$

Dually we define $FX \xrightarrow{\mathscr{P}^{\mathsf{opsim}}} FY$ to be $\{(u, v) \mid \forall (a, y) \in v.\ \exists x \in X.\ (a, x) \in u \wedge (x, y) \in \mathscr{R}\}$.

F lifts to an endofunctor G on **Preord** mapping $(X, \leqslant)$ to $(FX, \leqslant^{\mathsf{sim}})$ and we obtain an endofunctor H on **Poset** as the composite

$$\mathbf{Poset} \hookrightarrow \mathbf{Preord} \xrightarrow{G} \mathbf{Preord} \xrightarrow{Q} \mathbf{Poset}$$

Note that any F-coalgebra $A = (X, \theta)$ gives an H-coalgebra $\Delta A = (X, (=_X), \theta)$.

Now we relate $\lesssim$ to final H-coalgebras.

Proposition 3 (Using a final H-coalgebra)**.** *Fix a final H-coalgebra $\nu H = (P, \leqslant, \zeta)$, and for any H-coalgebra A we write $A \xrightarrow{c_A} \nu H$ for the unique anamorphism.*

Let $A = (X, \theta)$ and $B = (Y, \phi)$ be F-coalgebras, and let $x \in X$ and $y \in Y$. Then $x \lesssim_{A,B} y$ iff $c_{\Delta A} x \leqslant c_{\Delta B} y$.

Proposition 4 (Constructing a final F-coalgebra)**.** *Suppose that A is "big enough" up to mutual similarity, i.e. for every F-coalgebra $B = (Y, \phi)$ and $y \in Y$ there is $x \in X$ such that $x \lesssim_{A,B} y$ and $y \lesssim_{B,A} x$.*

Write Y for the poset $Q(X, \lesssim_{A,A})$ Then there is a unique monotone function $Y \xrightarrow{\phi} HY$ making $\Delta A \xrightarrow{p} (Y, \phi)$ a coalgebra morphism. Moreover (Y, ϕ) is final.

3 Nested simulation

Let A and B be coalgebras. A *2-nested simulation* [1] from A to B is a simulation contained in the converse of a simulation. We write $\lesssim^2_{A,B}$ for the largest 2-nested simulation.

A *nested preordered set* $(X, \leqslant_1, \leqslant_2)$ is a set with two preorders, where $\leqslant_2 \subseteq \leqslant_1$. It is a *nested poset* when $\leqslant_2$ is a partial order. We obtain categories

$$\mathbf{NestPoset} \underset{\top}{\overset{Q}{\hookleftarrow}} \mathbf{NestPreord}$$

F lifts to an endofunctor on **NestPreord** mapping $(X, \leqslant_1, \leqslant_2)$ to

$$(FX, \leqslant_1{}^{\mathsf{opsim}}, \leqslant_1{}^{\mathsf{opsim}} \cap \leqslant_2{}^{\mathsf{sim}})$$

and we obtain an endofunctor H on **NestPoset** as before.

Then $\lesssim^2$ can be characterized in terms of final H-coalgebras, and final H-coalgebras constructed using $\lesssim^2$, just as before.

4 General Theory

We require a notion of "fuzzy relation" between sets. For our 2-nested simulation example, a fuzzy relation $X \xrightarrow{\mathscr{R}} Y$ is a pair $(\mathscr{R}_1, \mathscr{R}_2)$ of relations with $\mathscr{R}_2 \subseteq \mathscr{R}_1$.

Let F be an endofunctor on **Set**. An *F-relator* maps each fuzzy relation $X \xrightarrow{\mathscr{R}} Y$ to another $FX \xrightarrow{\Gamma\mathscr{R}} FY$, with the following condition satisfied.

Monotonicity If $\mathscr{R} \subseteq \mathscr{S}$ then $\Gamma\mathscr{R} \subseteq \Gamma\mathscr{S}$.

Stability $\Gamma((f \times g)^{-1}\mathscr{R}) = (Ff \times Fg)^{-1}\Gamma\mathscr{R}$.

Lax functoriality $(=_{FX}) \subseteq \Gamma(=_X)$, and $(\Gamma\mathscr{R}); (\Gamma\mathscr{S}) \subseteq \Gamma(\mathscr{R}; \mathscr{S})$.

In [2, 3], strict preservation of binary composition is required. Our more liberal condition, which follows [4], allows 2-nested simulation as an example.

References

[1] Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

[2] Wim H. Hesselink and Albert Thijs. Fixpoint semantics and simulation. *Theor. Comput. Sci*, 238(1-2):275–311, 2000.

[3] Jesse Hughes and Bart Jacobs. Simulations in coalgebra. *Theor. Comput. Sci*, 327(1-2):71–108, 2004.

[4] James Worrell. Coinduction for recursive data types: partial orders, metric spaces and omega-categories. *Electr. Notes Theor. Comput. Sci*, 33, 2000.

Rigid structures and their application to the $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem

Tomasz Brengos
Warsaw University of Technology, Poland
t.brengos@mini.pw.edu.pl

Abstract

H. P. Gumm and T. Schröder stated a hypothesis that the preservation of preimages by a connected functor T is equivalent to the satisfaction of the class equality $\mathscr{H}\mathscr{S}(\mathsf{K}) = \mathscr{S}\mathscr{H}(\mathsf{K})$ for any class K of T-coalgebras. This talk will be devoted to the study of the relation between connected functors and the satisfaction of the equation $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$. We will discuss some interesting applications of different rigid structures to finding partial solution to the problem.

1 Introduction

H. P. Gumm and T. Schröder [3] stated the following problem: is the satisfaction of the equation $\mathscr{H}\mathscr{S}(\mathsf{K}) = \mathscr{S}\mathscr{H}(\mathsf{K})$ for any class K of T-coalgebras equivalent to T preserving preimages?

A positive answer to this question has been conjectured. The conjecture is supported by the following two theorems.

Theorem 1.1. *[3] Assume that a functor $T : \mathsf{Set} \to \mathsf{Set}$ preserves preimages. Then for any class K of T-coalgebras $\mathscr{H}\mathscr{S}(\mathsf{K}) = \mathscr{S}\mathscr{H}(\mathsf{K})$.*

Theorem 1.2. *[3] Let $T : \mathsf{Set} \to \mathsf{Set}$ be a functor such that $|T1| > 1$. If $\mathscr{H}\mathscr{S}(\mathsf{K}) = \mathscr{S}\mathscr{H}(\mathsf{K})$ for any class K of T-coalgebras then T preserves preimages.*

By Theorem 1.1 and Theorem 1.2 it follows that in order to find a positive answer to $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem it is enough to find, for any functor $T : \mathsf{Set} \to \mathsf{Set}$ not preserving preimages and satisfying $|T1| = 1$, a T-coalgebra $\mathbb{X}$ such that $\mathscr{S}\mathscr{H}(\mathbb{X}) \neq \mathscr{H}\mathscr{S}(\mathbb{X})$. However, the constant functor $\mathscr{C}_{0,1}$ sending the empty set to itself and all non-empty sets to a one-element set is a counterexample. The condition which should be investigated to be equivalent to $\mathscr{H}\mathscr{S}(\mathsf{K}) = \mathscr{S}\mathscr{H}(\mathsf{K})$ is the preservation of *non-empty preimages*.

Any Set-endofunctor T which does not preserve non-empty preimages and is *connected* (i.e. satisfies $|T1| = 1$) falls into exactly one of the two classes:

1. functors which do not preserve non-empty preimages on undistinguished points,

2. functors which preserve non-empty preimages on undistinguished points.

It can be shown that if a connected functor T not preserving non-empty preimages falls into the first class mentioned above then there is a T-coalgebra $\mathbb{X}$ for which $\mathscr{H}\mathscr{S}(\mathbb{X}) \neq \mathscr{S}\mathscr{H}(\mathbb{X})$. The construction of such coalgebra has been presented in [1] and will not be the topic of our talk. More interesting are the functors that fall into the second class. Unfortunatelly, the full solution to the $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem for these functors is not known. We will devote our talk to presenting partial solutions in which different rigid structures play an important role.

2 Functors preserving non-empty preimages on undistinguished points

Definition 2.1. A Set-endofunctor T which does not preserve non-empty preimages is said to *preserve non-empty preimages on undistinguished points* whenever for any mapping $f : A \to B$, any subset $C \subset B$ with $f^{-1}(C) \neq \emptyset$ and any undistinguished element $\vec{a} \in TA$ for which $Tf(\vec{a})$ is undistiguished, the membership $Tf(\vec{a}) \in TC$ implies that $\vec{a} \in T[f^{-1}(C)]$.

If a functor $T : \mathsf{Set} \to \mathsf{Set}$ preseves non-empty preimages on undistinguished points then the constant functor $\mathscr{C}_{0,1}$ is a proper subfunctor of T, i.e. $\mathscr{C}_{0,1} < T$. Moreover, if we additionally assume that T is connected then it can be shown (see [5] for details) that T contains exactly one copy of $\mathscr{C}_{0,1}$ as a subfunctor. We will focus only on connected functor preserving non-empty preimages on undistiguished points.

As it has already been mentioned, the full solution to the $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem for connected functor preserving non-empty preimages on undistiguished points is not known. The authors of [1] present a general setting in which the hypothesis is true and show that a wide class of connected functors preserving non-empty preimages on undistinguished points can be put into this setting. This class of functors is further extended in [2].

Theorem 2.2 ([1, 2])**.** *Let T be a connected Set-endofunctor preserving preimages on undistinguished points. If the functor T additionally satisfies at least one of the conditions:*

1. *there is a set A and an undistinguished element $\vec{a} \in TA$ such that the filter $\mathsf{Flt}_A(\vec{a})$ is not free,*
2. *$|T\omega| > 1$,*
3. *there is a set A of cardinality $\lambda > \omega$, an element $\vec{a} \in TA$ and a partition $\{A_l\}_{l\in\lambda}$ of the set A into λ-many non-empty subsets of A such that each of them is $\mathsf{Flt}_A(\vec{a})$-stationary,*
4. *there is an uncountable cardinal λ for which there is a countable family $\{S_i\}_{i\in\omega}$ of sets with $|S_i| < \lambda$ such that $\lambda = \bigcup_{i\in\omega} S_i$ and there is a set A of cardinality λ such that for each $i \in \omega$ there is an element $\vec{a}_i \in TA$ and a partition $\{A^i_j\}_{j\in S_i}$ of the set A into $|S_i|$-many non-empty disjoint $\mathsf{Flt}_A(\vec{a}_i)$-stationary subsets*

then there is a coalgebra $\mathbb{X}$ for which $\mathscr{H}\mathscr{S}(\mathbb{X}) \neq \mathscr{S}\mathscr{H}(\mathbb{X})$.

Although the last two conditions listed above seem to be very technical, they are the most interesting. The proof of the above theorem makes use of rigid structures when condition (3) or (4) are satisfied. Recall that given any category $\mathfrak{C}$, an object $A \in \mathfrak{C}$ is called *rigid* if its only endomorphism is the identity. If (3) is satisfied then we use a rigid directed graph of cardinality λ. If condition (4) holds then rigid unary algebra of cardinality λ is used. We would like to present in more detail the methods of putting these rigid structures into the coalgebraic setting and sketch the proof of Theorem 2.2 in the case the condition (3) or (4) holds.

References

[1] T. Brengos, V. Trnková, *On the $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem for coalgebras*, Algebra Universalis (to appear)

[2] T. Brengos, *Rigidity of unary algebras and its application to the $\mathscr{H}\mathscr{S} = \mathscr{S}\mathscr{H}$ problem*, preprint

[3] H. P. Gumm, T. Schröder, *Types and coalgebraic structure*, Algebra Universalis, no. 53, Numbers 2-3 , p. 229-252 (2005)

[4] H.P. Gumm, *From T-coalgebras to filter structures and transtion systems*, CALCO 2005, LNCS 3629, 2005.

[5] V. Trnková, *On Descriptive Classification of Set-functors I*, Commentationes Mathematicae Universitatis Carlinae 12, 1971

Positive structural operational semantics and monotone distributive laws

Marcelo Fiore and Sam Staton*
Computer Laboratory, University of Cambridge

Abstract

We describe a correspondence between (i) rule-based inductive definitions in the Positive GSOS format, i.e. without negative premises, and (ii) distributive laws in the spirit of Turi and Plotkin (LICS'97), that are suitably monotone. This result can be understood as an isomorphism of lattices, in which the prime elements are the individual rules.

Context

We fix a signature, i.e. a set S of operators, each $\mathrm{op} \in S$ having an arity, $\mathrm{ar}(\mathrm{op}) \in \mathbb{N}$. We also fix a set L of labels. We are concerned with labelled transition relations over the terms of the signature.

For an example, we recall a fragment of Milner's CCS. Fix a set A of actions. There is a binary operation for parallel composition (written $x \,|\, y$), a constant for deadlock, and two unary operations for every action $a \in A$, for prefix and co-action prefix.

The set of labels contains actions, co-actions, and silent steps: $L = A + A + 1$. A labelled transition relation over CCS terms is defined by a transition system specification (TSS). This is a set of rules, including the one in the box on the right.

$$\frac{x \xrightarrow{a} x' \quad y \xrightarrow{\bar{a}} y'}{x \,|\, y \xrightarrow{\tau} x' \,|\, y'}$$

Background on the Positive GSOS rule format

A transition system specification (TSS) over the signature is said to be in the GSOS format [2] if it comprises a set of rules of the form shown in the box on the right, where the variables are all distinct, and the variables in the term t already appear in the premises or on the left hand side of the conclusion. We will allow the sets J_i and K_i of premises to be infinite.

$$\frac{\{x_i \xrightarrow{l_{ij}} y_{ij} \mid i \leq \mathrm{ar}(\mathrm{op}), j \in J_i\} \quad \{x_i \not\xrightarrow{l_{ik}} \mid i \leq \mathrm{ar}(\mathrm{op}),\ k \in K_i\}}{\mathrm{op}(x_1, \ldots, x_{\mathrm{ar}(\mathrm{op})}) \xrightarrow{l} t}$$

The TSS is said to be in the Positive GSOS format if there are no negative premises: the sets K_i are all empty. The specification of CCS is in the Positive GSOS format. Aceto et al. [1] give an overview.

Background on distributive laws and mathematical operational semantics

We now recall some aspects of the 'mathematical operational semantics' of Turi and Plotkin [6]. The signature S induces an endofunctor Σ_S and a monad T_S on **Set**: for a set X, let $\Sigma_S(X) = \biguplus_{\mathrm{op} \in S} X^{\mathrm{ar}(\mathrm{op})}$, and let $T_S(X)$ be the set of terms built from the signature and the elements of X. We will also consider an endofunctor B_L on **Set**: for a set X, let $B_L(X) = \mathscr{P}(L \times X)$. (Here, $\mathscr{P}$ is the covariant powerset functor.) Coalgebras for B_L are in bijective correspondence with labelled transition systems.

We define a *distributive law* to be a family of functions $\rho_X \colon \Sigma_S(X \times B_L(X)) \to B_L(T_S(X))$, natural in $X \in$ **Set**. (Technically, this data defines a distributive law of a monad over a copointed endofunctor — see [4].) Every GSOS TSS gives rise to a distributive law, as follows. (For a set X, we define an *X-instance* of a rule to be a formal substitution of elements of X for variables in the rule.)

$$\rho_X\Big(\mathrm{op}\big((x_1,\beta_1),\ldots,(x_{\mathrm{ar}(\mathrm{op})},\beta_{\mathrm{ar}(\mathrm{op})})\big)\Big) = \left\{ (l,t) \;\middle|\; \begin{array}{l} \textit{there is an } X\textit{-instance of a rule in the TSS for which} \\ \textit{the conclusion is } \mathrm{op}(x_1,\ldots,x_{\mathrm{ar}(\mathrm{op})}) \xrightarrow{l} t\textit{, and for each} \\ \textit{premise } x_i \xrightarrow{l_{ij}} y_{ij}\textit{, the pair } (l_{ij},y_{ij}) \textit{ is in } \beta_i. \end{array} \right\} \quad (\dagger)$$

Proposition 1 (Turi and Plotkin)**.** *Every distributive law arises from a GSOS TSS.*

*Research supported by EPSRC grants GR/T22049/01 and EP/E042414/1.

Positivity and monotonicity

We now provide an analogue of Proposition 1 for the Positive GSOS format. For any set X, the set $\Sigma_S(X \times B_L(X))$ has a natural partial order, induced by the subset ordering of the powerset. If $\beta_1 \subseteq \beta_1', \ldots,$ and $\beta_{\mathrm{ar(op)}} \subseteq \beta'_{\mathrm{ar(op)}}$, then we let $\mathrm{op}((x_1,\beta_1),\ldots,(x_{\mathrm{ar(op)}},\beta_{\mathrm{ar(op)}})) \leq \mathrm{op}((x_1,\beta_1'),\ldots,(x_{\mathrm{ar(op)}},\beta'_{\mathrm{ar(op)}}))$. The set $B_L(T_S(X))$ can also be partially ordered by subset inclusion. We say that a distributive law $\{\rho_X : \Sigma_S(X \times B_L(X)) \to B_L(T_S(X))\}_{X \in \mathbf{Set}}$ is monotone if each function ρ_X is monotone with respect to the above orderings.

Theorem 2. *Every Positive GSOS TSS induces (via (†)) a monotone distributive law. Every monotone distributive law is induced by a Positive GSOS TSS.*

A lattice isomorphism

By a *model* of a Positive GSOS TSS, we understand a model of the first-order theory whose function symbols are the operators of the algebraic signature, together with a binary relation symbol $\xrightarrow{l}$ for each label $l \in L$, and a Horn clause for every rule in the TSS.

We say that two TSSs in the Positive GSOS format are equivalent if they have the same models. This accounts for a change in variables in the rules of a TSS, and for redundant premises. For instance, if we add the rule in the box on the right to the TSS for CCS, then the resulting TSS is equivalent.

$$\frac{v \xrightarrow{a} v' \quad w \xrightarrow{\bar{a}} w' \quad w \xrightarrow{\tau} w''}{v \,|\, w \xrightarrow{\tau} v' \,|\, w'}$$

The collection of equivalence classes of Positive GSOS TSSs (over our signature S) forms a set. We can equip this set with the following complete lattice structure. The join $\bigsqcup \mathscr{C}$ of a collection $\mathscr{C}$ of TSSs is the TSS that contains all the rules in all the TSSs in $\mathscr{C}$. This lattice is prime algebraic, and the TSSs that only contain one rule are the primes (viz. the TSSs that are inaccessible by joins).

The union structure of the powerset induces a pointwise join-semilattice structure on the collection of monotone distributive laws.

Theorem 3. *The construction in* (†) *defines an isomorphism between the complete lattice of Positive GSOS TSSs, and the complete lattice of monotone distributive laws.*

Mathematical operational semantics

This development suggests two abstract ideas in the context of mathematical operational semantics. Firstly, B_L can be replaced by an arbitrary functor **Set** $\to$ **Poset** (see [3]), and monotone distributive laws are nothing but natural transformations between functors **Set** $\to$ **Poset**. We can thus provide a categorical account of the following fact: *for a Positive GSOS TSS, similarity is a precongruence* (see also [5, §4]). Secondly, when B_L is a functor **Set** $\to$ **PrAlgLat** into prime algebraic lattices and join preserving maps, the primes in the lattice of monotone distributive laws are an abstract form of solitary rule.

References

[1] L. Aceto, W. Fokkink, and F. Vaandrager. Structural operational semantics. In *Handbook of Process Algebra*. Elsevier, 2001.

[2] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[3] J. Hughes and B. Jacobs. Simulations in coalgebra. *Theoret. Comput. Sci.*, 327:71–108, 2004.

[4] M. Lenisa, J. Power, and H. Watanabe. Distributivity for endofunctors, pointed and co-pointed endofunctors, monads and comonads. In *Proc. CMCS'00*, pages 230–260, 2000.

[5] S. Staton. General structural operational semantics through categorical logic. In *Proc. LICS'08*, 2008.

[6] D. Turi and G. D. Plotkin. Towards a mathematical operational semantics. In *Proc. LICS'97*, 1997.

Pointwise Extensions of GSOS-Defined Operations

Helle Hvid Hansen
Eindhoven University of Technology

Bartek Klin
University of Cambridge,
University of Warsaw

Distributive laws of syntax over behaviour (cf. [1, 3]) are, among other things, a well-structured way of defining algebraic operations on final coalgebras. For a simple example, consider the set B^ω of infinite streams of elements of B; this carries a final coalgebra $w = \langle hd, tl\rangle \colon B^\omega \to B \times B^\omega$ for the endofunctor $F = B \times -$ on **Set**. If B comes with a binary operation $+$, one can define an addition operation $\oplus$ on streams coinductively:

$$hd(\sigma \oplus \tau) = hd(\sigma) + hd(\tau) \qquad tl(\sigma \oplus \tau) = tl(\sigma) \oplus tl(\tau).$$

It is easy to see that these equations define a distributive law, i.e., a natural transformation $\lambda : \Sigma F \Rightarrow F\Sigma$, where $\Sigma X = X^2$ is the signature endofunctor corresponding to a single binary operation. The operation $\oplus \colon B^\omega \times B^\omega \to B^\omega$ is now defined as the unique morphism to the final coalgebra as in:

$$\begin{array}{ccccc} \Sigma B^\omega & \xrightarrow{\Sigma w} & \Sigma F B^\omega & \xrightarrow{\lambda_{B^\omega}} & F\Sigma B^\omega \\ {\scriptstyle\oplus}\downarrow & & & & \downarrow{\scriptstyle F\oplus} \\ B^\omega & & \xrightarrow[w]{\cong} & & FB^\omega \end{array} \tag{1}$$

For another example, consider an operation $\boxplus$ where the first element of one stream is added to every element of the other one. This can be defined by equations:

$$hd(\sigma \boxplus \tau) = hd(\sigma) + hd(\tau) \qquad tl(\sigma \boxplus \tau) = \sigma \boxplus tl(\tau).$$

These do not define a simple distributive law as above; however, they do define a slightly more complex law of the type $\lambda : \Sigma(\mathrm{Id} \times F) \Rightarrow F\Sigma$, and the operation $\boxplus$ is defined from it similarly as in (1).

Mealy machines are coalgebras for the functor $(B \times -)^A$, where A is the input, and B the output alphabet. A final coalgebra structure is carried by the set Γ of functions from A^ω to B^ω that are causal: An n-ary operation on streams is *causal* if the kth element of the result only depends on the first k elements of each of the arguments. Operations on B^ω can be pointwise extended to operations on functions from A^ω to B^ω in the obvious way, e.g., one can define $(x \bar{\oplus} y)(\sigma) = x(\sigma) \oplus y(\sigma)$ for $x, y \colon A^\omega \to B^\omega$. If an operation $* \colon (B^\omega)^n \to B^\omega$ is causal then the pointwise extension $\bar{*}$ preserves causality and hence is an operation on Γ.

Stream operations that are defined via a distributive law of syntax endofunctors Σ over $B \times -$, i.e., by the simple format used in the definition of $\oplus$, have pointwise extensions that are defined by a distributive law of Σ over $(B \times -)^A$. It follows that the resulting extended operations on functions preserve causality. Relevant distributive laws can be syntactically presented with a simple inference rule formalism. For example, the operation $\bar{\oplus}$ on Mealy machines is defined by rules:

$$\frac{x \xrightarrow{a|p} x' \quad y \xrightarrow{a|q} y'}{x \bar{\oplus} y \xrightarrow{a|(p+q)} x' \bar{\oplus} y'}$$

with a ranging over A and p, q ranging over B. The meaning of a clause $x \xrightarrow{a|b} x'$ is that a Mealy machine x, upon receiving input a, outputs b and transforms into a machine x'.

Pointwise extensions of some other operations are more problematic. For example, the naive idea to define the pointwise extension of $\boxplus$ by:

$$\frac{x \xrightarrow{a|p} x' \quad y \xrightarrow{a|q} y'}{x \bar{\boxplus} y \xrightarrow{a|(p+q)} x \bar{\boxplus} y'}$$

does not work, i.e., the resulting operation does not behave as the stream operation $\boxplus$ pointwise. Nevertheless, the pointwise extension of $\boxplus$ can be defined in terms of a distributive law, if one extends the syntax with a family of auxiliary operators. Specifically, for every $a \in A$ we add a unary operator $a \triangleright -$ and take rules:

$$\frac{x \xrightarrow{a|p} x'}{a \triangleright x \xrightarrow{b|p} b \triangleright x'} \qquad \frac{x \xrightarrow{a|p} x' \quad y \xrightarrow{a|q} y'}{x \bar{\boxplus} y \xrightarrow{a|(p+q)} (a \triangleright x) \bar{\boxplus} y'}$$

where a, b range over A and p, q over B. Intuitively, the new operators act like one-slot input buffers: a Mealy machine $a \triangleright x$, upon receiving input b, passes a as input to x, returns its output, and stores b in the buffer.

The above rules define a distributive law of the type $\lambda : \overline{\Sigma}(\mathrm{Id} \times \overline{F}) \Rightarrow \overline{F}\,\overline{\Sigma}^*$, where $\overline{F} = (B \times -)^A$, $\overline{\Sigma}$ corresponds to the syntax extended with auxiliary operators $a \triangleright -$, and $\overline{\Sigma}^*$ is the free monad over $\overline{\Sigma}$. Such laws are called (abstract) *GSOS laws* (cf. [1, 3]), and they induce operations on final F-coalgebras similarly as in (1). It turns out that the induced operation is the pointwise extension of the operation $\boxplus$ on streams.

So far we have described just two very simple examples of operations and their pointwise extensions. However, the techniques presented here are far more general: they apply to arbitrary operations definable with certain distributive laws, and even to arbitrary behaviour endofunctors F on **Set**. Specifically:

- Simple distributive laws $\lambda : \Sigma F \Rightarrow F\Sigma$ can be extended pointwise to laws $\overline{\lambda} : \Sigma(F-)^A \Rightarrow (F\Sigma-)^A$,
- GSOS distributive laws $\lambda : \Sigma(\mathrm{Id} \times F) \Rightarrow F\Sigma^*$ can be extended to laws $\overline{\lambda} : \overline{\Sigma}(\mathrm{Id} \times (F-)^A) \Rightarrow (F\overline{\Sigma}^*-)^A$, where $\overline{\Sigma}$ arises from Σ by adding a family of unary "buffer" operations $a \triangleright -$ for $a \in A$,

so that the resulting operations on final $(F-)^A$-coalgebras are pointwise extensions of their counterparts on F-coalgebras.

It is interesting that the technique of adding "buffer" operations is enough to solve the problem of pointwise extension for arbitrary GSOS operations and arbitrary behaviour functors. We would like to understand better the relationship of the buffer operations to the structure of the final $(F-)^A$-coalgebra. For example, in the case of streams and Mealy machines, for every causal function $f : A^\omega \to B^\omega$, $a \in A$ and $\sigma \in A^\omega$, we have that $(a \triangleright f)(\sigma) = f(a : \sigma)$, and the final $(B \times -)^A$-structure on Γ is the map $\gamma : \Gamma \to (B \times \Gamma)^A$ defined by $\gamma(f)(a) = \langle hd, tl \rangle \circ (a \triangleright f)$. Another observation is that the buffer operations are already expressible in Rutten's stream calculus (cf. [2]): if an expression $e(\sigma)$ is a stream calculus specification of a stream function f, then the expression obtained by substituting $[a] + X \times \sigma$ for σ in e specifies $a \triangleright f$, where $[a]$ denotes the stream $(a, 0, 0, 0, \ldots)$.

References

[1] F. Bartels. *On Generalised Coinduction and Probabilistic Specification Formats*. PhD thesis, Vrije Universiteit Amsterdam, 2004.

[2] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata and power series. *Theoretical Computer Science*, 308(1):1–53, 2003.

[3] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997)*, pages 280–291. IEEE Computer Society, 1997.

Dynamic Coalgebraic Modalities

Helle Hvid Hansen
Eindhoven University of Technology,
Centrum Wiskunde & Informatica, Amsterdam

Raul Andres Leal
Universiteit van Amsterdam

With this work we aim to place dynamic modal logics such as Propositional Dynamic Logic (PDL) [1] and Game Logic (GL) [4] in a uniform coalgebraic framework. In our view, a dynamic system $\mathbb{S}$ consists of the following ingredients:

1. A set S which represents the global states of $\mathbb{S}$.
2. An algebra L of labels (denoting actions, programs, games, ...).
3. An interpretation of labels as G-coalgebras on the state space S.
4. A collection of labelled modalities $[\alpha]$, for $\alpha \in L$, where intuitively $[\alpha]\varphi$ reads: "after α, φ holds".

Formally, the interpretation of labels is a map $\sigma\colon L \to (GS)^S$ which describes how actions change the global system state. The algebraic structure on L describes how one can compose actions into more complex ones. The same type of algebraic structure should be carried by $(GS)^S$, and we say that σ is *standard*, if σ is an algebra homomorphism, which means that the semantics of actions is compositional. By considering the exponential adjoint $\widehat{\sigma}\colon S \to (GS)^L$ we obtain a behavioural description of the system in the form of a G^L-coalgebra. These two (equivalent) views of a dynamic system form the basis of our modelling. In short, σ describes structure and dynamics, and $\widehat{\sigma}$ describes behaviour and induces modalities.

$$\sigma\colon L \to (GS)^S \qquad\qquad \widehat{\sigma}\colon S \to (GS)^L$$
(algebraic view: structure, dynamics) (coalgebraic view: behaviour, modalities)

A similar observation was made in [2] in the context of Java semantics, which we will return to later.

PDL (without tests) can be seen as an instance of the above by taking as L the set of program expressions, and as G the covariant powerset functor $\mathscr{P}$. The algebraic structure on $(\mathscr{P}S)^S$ is given by the operations on relations that define a standard PDL frame (cf. [1]). Similarly, GL (without tests) is obtained by taking G to be the monotonic neighbourhood functor $\mathscr{M}$. Observe that the PDL-modalities are just labelled versions of the usual $\mathscr{P}$-modality $\Box$. Generalising this construction, we obtain G^L-modalities by labelling G-modalities. More precisely, given $\alpha \in L$ and a predicate lifting $\lambda\colon \mathscr{Q} \to \mathscr{Q}G$ (where $\mathscr{Q}$ denotes the contravariant power set functor), we define the *α-labelling of λ* to be the predicate lifting with S-component

$$\lambda^\alpha_S\colon \mathscr{Q}S \to \mathscr{Q}(GS)^L; \quad U \mapsto \lambda^\alpha_S(U) = \{\delta \in (GS)^L \mid \delta(\alpha) \in \lambda_S(U)\}$$

Also the modalities in Game Logic and Java semantics (cf. [2]) arise as labelled modalities. Every predicate lifting $\lambda^\alpha \in \Lambda^L_G$ induces a predicate transformer $m_{\lambda^\alpha} = \sigma(\alpha)^{-1} \circ \lambda_S\colon \mathscr{Q}S \to \mathscr{Q}S$ on a system $\mathbb{S}$, and the truth set of a modal formula $[\alpha]_\lambda(\phi)$ in $\mathbb{S}$ is defined as $[\![[\alpha]_\lambda(\phi)]\!] = m_{\lambda^\alpha}([\![\phi]\!])$.

The structure of standard PDL frames is axiomatised by formulas such as $[\alpha;\beta]\phi \leftrightarrow [\alpha][\beta]\phi$ (SEQ) and $[\alpha \cup \beta]\phi \leftrightarrow [\alpha]\phi \wedge [\beta]\phi$ (CHOICE). These axioms display an interaction between the $\Box$-modality for $\mathscr{P}$ and the algebraic structure on programs. We would like to understand such axioms in our abstract setting. Our focus so far has been on the axiom (SEQ) for sequential composition. It is rather well known (see e.g. [3]) that sequential composition can be more generally understood as Kleisli composition for a monad. Indeed, for PDL (resp. GL) we have that $\sigma(\alpha;\beta) = \sigma(\alpha) * \sigma(\beta)$, where $*$ denotes Kleisli composition for $\mathscr{P}$ (resp. $\mathscr{M}$). Hence if G is a monad, then an operation ; can be defined on $(GS)^S$ as Kleisli composition for G. Since $[\alpha]_\lambda$ is interpreted by a predicate lifting λ^α for G^L, (SEQ) should be read parametric in the underlying predicate lifting λ. This is made explicit in the following formulation of (SEQ) in terms of predicate transformers.

Definition 1. A predicate lifting $\lambda : \mathscr{Q} \to \mathscr{Q}G$ for a monad G *captures sequential composition* if for all standard $\sigma : L \to (GS)^S$ and all $\alpha, \beta \in L$: $m_{\lambda^{\alpha;\beta}} = m_{\lambda^\alpha} \circ m_{\lambda^\beta}$.

We point out that Definition 1 is not vacuous. For example, the constant predicate lifting λ for $\mathscr{P}$ with value $\lambda_S(U) = \{\emptyset\}$ for all $U \subseteq S$ does not capture sequential composition. To see this, for a state $s \in S$ we have: $s \in m_{\lambda^{\alpha;\beta}}(U)$ iff $\sigma(\alpha;\beta)(s) = \emptyset$, while $s \in m_{\lambda^\alpha}(m_{\lambda^\beta}(U))$ iff $\sigma(\alpha)(s) = \emptyset$. Hence the inclusion from left to right fails in general. (Take for example $\sigma(\alpha) = S \times S$ and $\sigma(\beta) = \emptyset$.) A second (easy) observation is that λ captures sequential composition iff its boolean dual $\neg\lambda\neg$ does. Our main result up to now is the following characterisation.

Theorem 2. Let G be a monad. A predicate lifting $\lambda : \mathscr{Q} \to \mathscr{Q}G$ captures sequential composition iff its adjoint $\widehat{\lambda} : G \to \mathscr{Q}\mathscr{Q}$ is a monad morphism.

From this theorem, we can reprove the validity of (SEQ) for Game Logic: If $\lambda : \mathscr{Q} \to \mathscr{Q}\mathscr{M}$ is the predicate lifting for the monotonic box, then $\widehat{\lambda} : \mathscr{M} \to \mathscr{Q}\mathscr{Q}$ is simply the natural inclusion which is a monad morphism. Note that checking whether a natural transformation is a monad morphism is an algorithmic procedure.

Our dynamic systems can be seen as a special case of the framework presented in [2] to give semantics to Java programs. The main difference with PDL and GL is that Java programs manipulate data as well as effecting state change. In [2], a Java program with input in A and output in B is formalised as a map $A \to F(S,B)^S$ where $F : \mathit{Set} \times \mathit{Set} \to \mathit{Set}$ is a bifunctor such that for a fixed S, the functor $F(S,-)^S$ is a monad. In particular, sequential composition is Kleisli composition for $F(S,-)^S$. PDL is obtained in this framework by taking $A = B = 1$ (PDL programs have trivial input/output), and $F(S,B) = \mathscr{P}(B \times S)$. Taking the exponential adjoint, a Java program is also a coalgebra $S \to F(S,B)^A$, and thus gives rise to labelled $F(-,B)^A$-modalities and predicate transformers. However, since Java program operations are subject to typing conditions, and hence not totally defined, a standard interpretation can no longer be defined as an algebra homomorphism, and the collection of all programs does not form a labelled $F(-,B)^A$-coalgebra. Still we can generalise the notion of standardness and Definition 1 such that Theorem 2 still holds.

There are many questions we would like to address in the future. We have seen that if G is a monad, then $(GS)^S$ supports a notion of sequential composition. We would like to investigate the general relationship between properties of the functor G, operations supported by $(GS)^S$, and compositionality axioms. We also would like to develop new examples of dynamic modalities involving probabilistic systems. An issue that we left open was how to include the program/game construction of tests in our framework.

Acknowledgements We thank Bart Jacobs for suggesting this topic to us, and both Bart Jacobs and Yde Venema for stimulating discussions and helpful comments.

References

[1] R. Goldblatt. *Logics of Time and Computation*. Number 7 in CSLI Lecture Notes. Center for the Study of Language and Information, 2nd edition, 1992.

[2] B. Jacobs and E. Poll. Coalgebras and monads in the semantics of Java. *Theoretical Computer Science*, 291:329–349, 2003.

[3] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[4] R. Parikh. The logic of games and its applications. In *Topics in the Theory of Computation*, number 14 in Annals of Discrete Mathematics. Elsevier, 1985.

Conway Games, algebraically and coalgebraically

Furio Honsell
University of Udine, Italy.
furio.honsell@comune.udine.it

Marina Lenisa
University of Udine, Italy.
lenisa@dimi.uniud.it

In this talk, we will report on the most recent developments of our investigation on Conway games, which we started in [3].

This work arises from our attempt of understanding *games* using a very foundationally unbiased tool, namely *algebraic* and *coalgebraic methods*. Games arising in the literature and in real life are extremely varied. They exhibit indeed a perfect example of a *family resemblance* in the sense of Wittgenstein. Furthermore, in the past decades people have gone into the habit of describing, more or less conveniently, an extremely wide gamut of interactions and other dynamic phenomena using game-based *metaphors*. To make matters even more complex, when we speak about games many related *concepts* and *notions* come about, *e.g. move*, *position*, *play*, *turn*, *winning condition*, *payoff function*, *strategy*. None of these has a universal unique meaning, and according to the various presentations, these concepts are often blurred, sometimes taken as primitive sometimes explained and reduced to one another. And there are many more *properties*, which need to be specified more or less informally before actually having pinned down the kind of game one is interested in; *e.g. perfect knowledge*, *zero-sum*, *chance*, *number* of players, *finiteness*, *determinacy*.

We think that Conway's approach to games provides a very elementary and sufficiently abstract notion of game, which nonetheless is significantly structured, because of the special role that *sums* of games have in Conway's theory. And algebraic-coalgebraic methods provide a convenient conceptual setting for addressing this key concept.

For these reasons, in this talk, we focus on Conway games [2], that is *combinatorial games*, namely no chance 2-player games, the two players being conventionally called *Left* (L) and *Right* (R). Such games have *positions*, and in any position there are rules which restrict L to move to any of certain positions, called the *Left positions*, while R may similarly move only to certain positions, called the *Right positions*. L and R move in turn, and the game is of *perfect knowledge*, *i.e.* all positions are public to both players. The game ends when one of the players has no move, the other player being the winner, the *payoff* function yielding only 0 or 1. Many games played on boards are combinatorial games, *e.g. Nim*, *Domineering*, *Go*, *Chess*. Games, like Nim, where for every position both players have the same set of moves, are called *impartial*. More general games, like Domineering, Go, Chess, where L and R may have different sets of moves are called *partizan*.

Many other notions of games such as those which arise either in Set Theory, or in Automata Theory, or in Semantics of Programming Languages can be conveniently encoded in Conway's format.

In this talk, after a discussion addressing some general concepts and choices arising in different game presentations, we will outline and revisit Conway's theory of games and winning strategies under an *algebraic perspective*. Then, we will focus on *hypergames*, *i.e.* potentially infinite games, as introduced in [3]. Especially in view of applications, potentially infinite interactions are even more important than finite ones. In this respect we take the natural view that all infinite plays are *draws*. In the present talk, in fact, we shall address only games were all infinite plays are draws, *i.e.* "free" games in Conway's terminology; we will not deal with situations where certain infinite games are set to be winning for one of the two players, namely "fixed" (were infinite plays are all winning for one of the two players) or "mixed" games in Conway's terminology, see *e.g.* [1] for more details in this direction.

Initially, in [2], the author focussed only on *finite, i.e. terminating games*. Infinite games were intentionally neglected as ill-formed or trivial games, not interesting for "busy men", and their discussion

was essentially confined to a single chapter. Infinity (or *loopy*) games have been later considered in [1], Chapters 11-12, were free, fixed and mixed games have been considered. In these chapters, the authors also consider an interesting generalization of the Grundy-Sprague theory, due to Smith, which provides natural notions of *canonical forms*. However, not much attention has yet been paid to generalize the approach in [2] to infinite free games. This is what we have tried to do in [3].

In this talk, we will present a coalgebraic account of games, which is very natural and paves the way to a smooth and insightful treatment of infinite games. It allows us to consider games up-to bisimilarity, and to generalize operations and relations on them as congruences up-to bisimilarities. Moreover, the coalgebraic setting makes explicit the common nature between processes and games. For hypergames the notion of *winning strategy* has to be replaced by that of *non-losing strategy*, since we take non terminating plays to be draws. Hypergames can be naturally defined as a *final coalgebra* of *non-wellfounded sets* (*hypersets*), which are the sets of a universe of Zermelo-Fraenkel satisfying a suitable Antifoundation Axiom. Our theory of hypergames generalizes the original theory on finite games of [2] rather smoothly, but significantly. Two important results are a *Determinacy* and a *Characterization Theorem* of non-losing strategies on hypergames. The latter requires (a non-trivial) generalization of Conway's partial order relation on games to hypergames. Once hypergames are defined as a final coalgebra, operations on games, such as *disjunctive sum*, can be naturally extended to hypergames, by defining them as *final morphisms* into the coalgebra of hypergames.

We will discuss *Equivalences* and *congruences* on games and hypergames, providing in particular various characterizations of the *greatest congruence* w.r.t. sum, refining the *equideterminacy* relation. We will also discuss the special class of impartial hypergames. In particular, we will revisit and extend in a coalgebraic setting the theory of Grundy-Sprague and Smith based on the canonical *Nim games*, by introducing suitable *canonical ∞-hypergames*. We will show that such canonical hypergames can be construed in our setting as a truly *compositional semantics* of impartial hypergames, *fully abstract* w.r.t. the greatest behavioral congruence. Such semantics is given via a suitable *generalized Grundy function*, which we define on the whole class of hypergames. Our approach extends other approaches in the literature, where the generalized Grundy function is defined only on certain classes of finite cyclic graphs. Finally, we will discuss a normalizing procedure leading to canonical form.

Some References

[1] E. Berlekamp, J. Conway, R. Guy. Winning Ways, Academic Press, 1982.

[2] J.H. Conway. On Numbers and Games, second edition, A K Peters Ltd, 2001 (first edition by Academic Press, 1976).

[3] F. Honsell, M. Lenisa. Conway Games, coalgebraically, CALCO'09 Proc., Springer LNCS, vol. 5728, 2009, 300-316. Extended version available at "http://sole.dimi.uniud.it/∼marina.lenisa/Papers/Soft-copy-pdf/calco09extended.pdf"

Expressivity of Coalgebraic Logic over Posets

Krzystof Kapulkin
University of Warsaw
Poland
k.kapulkin@gmail.com

Alexander Kurz
Department of Computer Science
University of Leicester
United Kingdom
kurz@mcs.le.ac.uk

Jiří Velebil*
Faculty of Electrical Engineering
Czech Technical University in Prague
Czech Republic
velebil@math.feld.cvut.cz

1 Introduction

Most of coalgebraic logic so far has focussed on set-coalgebras and their Boolean logics. Exceptions include [7], [5], [1] and the work on coalgebras over measurable spaces [2]. Here we start a systematic investigation of coalgebras over posets and set ourselves the modest aim to show how to transfer to Pos the result of Schröder [8] and Klin [4], namely that the logic of all predicate liftings is expressive.

The basic ingredients of Schröder's argument [8] are still available in Pos, if we replace the discrete two-element set 2 by the two-element linear order $\mathbb{2}$. This means that formulas now correspond to upsets and expressivity means not only that non-bisimilar points can be separated, but also that the order between non-bisimilar points can be recovered from the logical information.

Although we do not emphasize this point of view, the proper framework in which to develop this generalisation is that of enriched category theory. Since categories enriched over Pos are simply categories with ordered hom-sets we do not need to go into the technicalities of enriched category theory. It only appears in justifying the crucial use of the Yoneda lemma and the assumption that our functors $T : \mathsf{Pos} \to \mathsf{Pos}$ are locally monotone (i.e., they are enriched functors).

The point where, compared to [8], a new phenomenon enters the picture is that Set but not Pos has the special property that every monomorphism with a non-empty domain is preserved by any functor. Monomorphisms of course play a special role, because expressivity is about separating (non-bisimilar) points and the crucial technical step in an expressivity argument (see [5] and [3]) typically involves a class of monos preserved by T. In our case, it will be *embeddings*, ie order-reflecting maps, since we want that the semantic order on coalgebras is characterised by the logical order. Thus, the expressivity result will need that functors $T : \mathsf{Pos} \to \mathsf{Pos}$ preserve embeddings.

We believe that logics of coalgebras over Pos are interesting for several reasons. First, from the point of view of coalgebraic logic, Pos is one of the obvious choices to consider, if one is interested in investigating the inherent generality in the notion of a coalgebra over a base category. Second, from the point of view of Stone duality, the adjunction between Pos and DL (distributive lattices) shares all the basic properties with the adjunction Set and BA (Boolean algebras), which promises that many further results can be obtained in analogy with the Set-based setting. Third, we see coalgebras over Pos as a bridge between set-coalgebras and domain theory (ie, coalgebras over domains), thus being a step towards a more systematic connection between the two areas. Last but not least, many of the techniques used in the Stone duality approach to coalgebraic logic are available in the setting of enriched category theory, a line of research that will be pursued in the future.

*The support of the grant MSM6840770014 of the Ministry of Education of the Czech Republic is acknowledged.

2 Results

Technically, in this paper, we replace the usual contarvariant adjunction between Set and BA of diagram by an adjunction

$$\mathsf{Pos}^{op} \underset{Pred}{\overset{Stone}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathsf{DL} \tag{1}$$

between Pos and DL, the category of distributive lattices. The above adjunction is to be considered as an adjunction in the *enriched sense*. This means that both the predicate functor *Pred* and the Stone functor *Stone* are locally monotone and that there is an isomorphism $\mathsf{Pos}^{op}(StoneA, X) \cong \mathsf{DL}(A, PredX)$ of *posets*, natural in X and A.

The predicate functor *Pred* assigns the poset $\mathsf{Pos}(X, \mathbb{2})$ endowed with the canonical structure of a distributive lattice to a poset X. In elementary terms, *PredX* is the distributive lattice of upper-sets in X.

The Stone functor $Stone : \mathsf{DL} \to \mathsf{Pos}^{op}$ assigns the poset $\mathsf{DL}(A, \mathbb{2})$ of to the distributive lattice A.

Given a locally monotone $T : \mathsf{Pos} \to \mathsf{Pos}$, the general theory allows us to define canonically (by mimicking exactly [5, 6]) a pair (L, δ) called *logic*, consisting of a locally monotone functor $L : \mathsf{DL} \to \mathsf{DL}$ and a natural transformation $\delta : LPred \to PredT^{op}$. By general facts about adjunctions, the transformation δ has a uniquely determined *mate* $\tau : StoneL \to T^{op}Stone$.

Recall that a logic (L, δ) is called *expressive* if states that are logically indistinguishable are already behaviourally equivalent.

Theorem 1. *Consider a locally monotone functor $T : \mathsf{Pos} \to \mathsf{Pos}$ preserving embeddings (ie order-reflecting maps) and a logic (L, δ) for T. If the τ_X are embeddings, then the logic is expressive.*

One can define predicate lifings analogously to the set-case and from the (enriched version of) Yoneda lemma it follows that n-ary predicate liftings form a poset $\mathsf{Pos}(T([n, \mathbb{2}]), \mathbb{2})$. Here, one may allow *any* finite poset n to be an arity and $[n, \mathbb{2}]$ is then a poset of all monotone maps from n to $\mathbb{2}$. This greater freedom of choice of arities is due to enrichment. We can, however, prove that expressivity is obtained even if one restricts arities to discrete finite posets.

Theorem 2. *Let $T : \mathsf{Pos} \to \mathsf{Pos}$ be a finitary and locally monotone functor preserving embeddings. The logic of all predicate liftings (with discrete arities) is expressive.*

References

[1] M. Bonsangue and A. Kurz. Pi-calculus in logical form. In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007)*, pages 303–312. IEEE, 2007.

[2] E.-E. Doberkat. *Stochastic Coalgebraic Logic*. Springer, 2010.

[3] B. Jacobs and A. Sokolova. Exemplaric expressivity of modal logics. *J. Logic Computation*. To appear.

[4] B. Klin. The least fibred lifting and the expressivity of coalgebraic modal logic. In *CALCO'05*.

[5] B. Klin. Coalgebraic modal logic beyond sets. In *MFPS'07*, 2007.

[6] A. Kurz and J. Rosický. The Goldblatt-Thomason-theorem for coalgebras. In *CALCO'07*.

[7] A. Palmigiano. A coalgebraic view on positive modal logic. *Theoret. Comput. Sci.*, 327, 2004.

[8] L. Schröder. Expressivity of Coalgebraic Modal Logic: The Limits and Beyond. In *FoSSaCS'05*.

Interpretations as coalgebra morphisms

Manuel A. Martins
Dep. Mathematics
University of Aveiro
martins@ua.pt

Alexandre Madeira
Dep. Informatics, U. Minho
Dep. Mathematics, U. Aveiro
Critical Software S.A.
madeira@ua.pt

Luis S. Barbosa
Dep. Informatics & CCTC
University of Minho
lsb@di.uminho.pt

1 Motivations

The concept of *logic or deductive system* is transversal in computer science. Several notions have been proposed on the literature, some of them, highly abstract in order to capture a wide class of logics. An interesting one is the formalization of a logic as an algebra together with a closure operator over that universe (or, equivalently, as an algebra together with a closure system over its universe (eg. [2])). Palmigiano in [6] related this *theory of abstract logic* with the *theory of coalgebras*, another central field in computer science. She shows that an abstract logic can be represented as a $\mathscr{C}$-coalgebra over **Set** with $\mathscr{C}$ the *closure system contravariant functor*; moreover the coalgebraic morphisms are the strict morphisms between the correspondent original logics, i.e., algebraic morphisms such that the pre-images of closed sets of the target logic are exactly the closed sets of the source logic.

In the context of algebraic specification refinement, we came recently interested in relating "equivalent logics" by maps which fail to be signature morphisms, in particular, in [3] and [4], we have studied how refinements can be witnessed by logical interpretation. A logical interpretation is a multifunction f that preserves consequence in the following sense: given two logics $\mathscr{A} = \langle A, C_A \rangle$ and $\mathscr{B} = \langle B, C_B \rangle$, for all $\{x\} \cup X \subseteq A$, $x \in C_A([X])$ iff $f(x) \subseteq C_B(f[X])$. A paradigmatic example of an interpretation, is multifunction $\tau : \mathrm{Fm}(\mathrm{Bool}) \to \mathrm{Fm}(\mathrm{CPC})$ from boolean equations to propositional terms defined, for any $\varphi \approx \varphi' \in \mathrm{Fm}(\mathrm{Bool})$, by $\tau(\varphi \approx \varphi') = \{\varphi \to \varphi', \varphi' \to \varphi\}$. The aim of the present work is to frame logics and interpretations along the lines of the coalgebraic perspective forward in [6].

2 The category of logics and interpretations and category $Coalg(\mathscr{C})$

The *category of logics and interpretations* is defined as $\langle \mathbf{Log}, \mathbf{Int}, \mathbf{i}, \circ \rangle$, where **Log** is the class of abstract logics, **Int** the class of its interpretations, **i** the class of identical maps (for each abstract logic $\mathscr{A}$, a multifunction $i_{\mathscr{A}}(x) = \{x\}$) and $\circ$ is the usual composition of multifunctions. We show that this category has a strong relation with the category $Coalg(\mathscr{C})$ on **Pw**, in which the objects are $\{\mathscr{P}(X) | X \in Obj(\mathbf{Set})\}$ and, the arrows, the functions between them. Namely, all abstract logics are $\mathscr{C}$-coalgebras and, its interpretations the respective coalgebraic morphisms.

A *minimal logic of* $\mathscr{A} = \langle A, C_A \rangle$ is defined as the logic $\mathscr{A}^* = \langle A_\sim, C_{\sim_{\mathscr{A}}} \rangle$ where $\sim$ is the equivalence relation which identifies elements with the same consequence, and $C_{\sim_{\mathscr{A}}}$, the closure operator coinduced by C_A. Interesting considerations can be proved concerning these minimizations: canonical projections with respect to $\sim$ are interpretations; moreover, given two abstract logics, there exists an interpretation between them if and only if there is an interpretation between the respective minimizations. This construction may overcame the pairwise redundancy on the axiomatizations of logics, since two properties with equal consequence are identified, via the canonical projection, in its minimization. This aspect seems to be useful for the software specification process where, for instance, the redundancy of proof obligations is naturally undesirable. Similar results are proved in [1] in the context of the theory of the conservative translations of logics.

3 Categorical approach to refinement via interpretation

Stepwise refinement of algebraic specifications is a well known formal methodology for software development. However, traditional notions of refinement based on signature morphisms are often too rigid to capture a number of relevant transformations in the context of software design, reuse, and adaptation. Based on this assumptions the authors suggested in [3, 4] an alternative notion of specification refinement, called *refinement via interpretation* where translations induced by signature morphisms are replaced by logical interpretations. The idea of this formalization is based on the existence of a largest interpretation: given a multifunction suitable to interpret a specification, there is a largest interpretation of that specification (i.e., a specification with the largest class of models) which may be defined as the largest specification whose models satisfy the translation of all properties that the refined specification satisfies. Based on this fact, we take as admissible refinements all the strict refinements of those interpretations.

Considering specifications as abstract logics and approaching consequences as coalgebras, an elegant view of specification refinement can be achieved. Namely, the *category of specifications and strict refinements* is defined and a categorical formalization of refinement via interpretation is presented. Morphisms of this category, the strict refinements, are *forward morphisms* in sense of [5], more precisely, inclusion forward morphisms with respect to the set-theoretic inclusion. In this context we characterize refinement via interpretation as follows: specification SP' refines SP if there is a commuting diagram

$$\begin{array}{ccccccccc}
A & \xrightarrow{int'} & \cdots & \xrightarrow{int} & B & \xrightarrow{ref} & \cdots & \xrightarrow{ref'} & B \\
\downarrow{\scriptstyle SP} & & & & \downarrow & & & & \downarrow{\scriptstyle SP'} \\
\mathscr{C}A & \xleftarrow[\mathscr{C}(int')]{} & \cdots & \xleftarrow[\mathscr{C}(int)]{} & \mathscr{C}B & \xleftarrow[\mathscr{C}(ref)]{} & \cdots & \xleftarrow[\mathscr{C}(ref')]{} & \mathscr{C}B
\end{array}$$

where the left and right squares commute in the category of specifications and interpretations and in the category of specifications and refinements respectively. This result naturally holds since, the morphisms of the mentioned categories are interpretations and strict refinements.

References

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

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

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

[4] M. A. Martins, A. Madeira, and L. S. Barbosa. Refinement via interpretation. In D. V. Hung and P. Krishnan, editors, *SEFM*, pages 250–259. IEEE Computer Society, 2009.

[5] S. Meng and L. S. Barbosa. Components as coalgebras: The refinement dimension. *Theor. Comput. Sci.*, 351(2):276–294, 2006.

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

Recursive Program Schemes with Effects

Daniel Schwencke
Technische Universität Braunschweig, Germany
schwencke@iti.cs.tu-bs.de

1 Introduction

Recursive program schemes (RPSs) are a useful tool for defining new operations from existing ones using recursion. A category theoretic approach and generalisation of RPSs has been given by S. Milius and L. S. Moss [MM06]. Building upon their work we present an approach to RPSs with additional effects.

Our leading example effect is non-determinism: non-deterministic RPSs have already been investigated by A. Arnold and M. Nivat [AN77] more than 30 years ago; the recursive definitions of such RPSs may additionally involve non-deterministic choice, or equivalently, the defined operations may have sets of results. For example, given a binary operation $\cdot$, one can define a unary operation *pow* by the non-deterministic RPS $pow(x) \approx \{x, x \cdot pow(x)\}$. Further effects include partiality, probabilism or environments; to our knowledge RPSs with these effects have not yet been investigated.

The aim of our work (in progress) is to prove category theoretic (as far as possible) that so called guarded RPSs with these effects define operations uniquely. Below we propose definitions of RPSs with effects, guardedness and (uninterpreted) solutions of such RPSs; the latter formalise what it means for RPSs to "define an operation". We briefly sketch our progress, the technical difficulties that arise and conclude with ideas for future work.

2 Preliminaries

We assume M to be a monad on **Set**. This monad captures the effect, e. g. for non-determinism one would take the powerset monad $M = \mathscr{P}$. Furthermore, we assume H and V to be polynomial **Set**-functors according to the signature of the existing operations and the operations we want to define, respectively. We require both H and V to have distributive laws $\lambda : HM \to MH$ and $\nu : VM \to MV$ over the monad M. Observe that this yields a distributive law $\rho = [M\mathsf{inl}, M\mathsf{inr}] \cdot (\lambda + \nu)$ of $H + V$ over the monad M. These distributive laws are needed to extend operations to arguments with effects.

For a polynomial endofunctor G we denote by (F^G, η^G, μ^G) the free monad on G with universal transformation $\kappa^G : G \to F^G$. We introduce the notation $\phi^G = \mu^G \cdot \kappa^G F^G : GF^G \to F^G$. For a natural transformation $\sigma : G \to T$ into a monad T we denote by $\sigma^\# : F^G \to T$ the unique monad morphism such that $\sigma^\# \cdot \kappa^G = \sigma$.

3 Definitions and Results

Lemma 3.1. *Let G be an endofunctor that has free algebras. Then any distributive law δ of G over the monad M induces a distributive law $\delta' : F^G M \to MF^G$ of monads.*

In particular this means that the composite $(MF^G, \eta^M F^G \cdot \eta^G, \mu^M F^G \cdot MM\mu^G \cdot M\delta' F^G)$ is a monad, see [Bec69].

Definition 3.2. A *RPS with M-effects* is a natural transformation $e : V \to MF^{H+V}$. It is called *guarded* if it factors through some $e_0 : V \to M(HF^{H+V} + \mathrm{Id})$, i. e.

$$e \equiv V \xrightarrow{e_0} M(HF^{H+V} + \mathrm{Id}) \xrightarrow{\mathrm{M(inlF^{H+V} + Id)}} \mathrm{M((H+V)F^{H+V} + Id)} \xrightarrow{\mathrm{M}[\phi^{\mathrm{H+V}}, \eta^{\mathrm{H+V}}]} \mathrm{MF^{H+V}} .$$

An *(uninterpreted) solution* of e is a natural transformation $e^\dagger : V \to MF^H$ such that $e^\dagger = \mu^M F^H \cdot M[\eta^M F^H \cdot \kappa^H, e^\dagger]^\# \cdot e$.

For several monads M including the maybe monad $\_ + 1$ (partiality), powerset monad $\mathscr{P}$ (non-determinism) and subdistribution monad $\mathscr{D}$ (probabilism), we would like to prove the following

Theorem 3.3. *Every guarded RPS e with M-effects has a unique (uninterpreted) solution.*

To this end, following [MM06], we define the endofunctor $\mathscr{H} = H \cdot \_ + \mathrm{Id}$ on the functor category $[\mathbf{Set}, \mathbf{Set}]$. We can extend the RPS e to $\bar{e} = [\eta^M (HF^{H+V} + \mathrm{Id}) \cdot \mathsf{inl} \cdot \mathrm{H}\eta^{\mathrm{H+V}}, \mathrm{e}_0]^\# : \mathrm{F}^{\mathrm{H+V}} \to \mathrm{M}(\mathrm{HF}^{\mathrm{H+V}} + \mathrm{Id}) = \mathrm{M}(\mathscr{H}\mathrm{F}^{\mathrm{H+V}})$. Observe that the codomain of $\bar{e}$ is indeed a monad since $\mathscr{H}$ is a functor on a certain category of monads, see loc. cit., and we obtain a distributive law of $\mathscr{H}F^{H+V}$ over the monad M using λ and ρ'. $\bar{e}$ performs second order substitution (according to e) in arbitrary terms built from the operations from H and V.

Additionally, from M we can form a monad $(\mathscr{M} = M \cdot \_, \eta^M{}_\_, \mu^M{}_\_)$ on $[\mathbf{Set}, \mathbf{Set}]$. Using λ, we obtain a distributive law of $\mathscr{H}$ over the monad $\mathscr{M}$ in $[\mathbf{Set}, \mathbf{Set}]$, or equivalently [Mul94], a lifting $\bar{\mathscr{H}}$ of $\mathscr{H}$ to the Kleisli category $[\mathbf{Set}, \mathbf{Set}]_{\mathscr{M}}$.

However, one cannot adapt the proof of [MM06], where such a theorem was proven for RPSs without effects, smoothly to RPSs with effects. The main technical difficulties that arise are (1) the final $\bar{\mathscr{H}}$-coalgebra $\bar{c}$ must be a lifting of a certain $\mathscr{H}$-coalgebra and (2) the unique $\bar{\mathscr{H}}$-coalgebra homomorphism from $\bar{e}$ to $\bar{c}$ must be a monad morphism.

For the three monads M mentioned above we can exploit the property that $\mathbf{Set}_M$ is CPO-enriched with strict composition and the lifting $\bar{H}$ of H induced by certain strict distributive laws λ is locally continuous. This solves (1); we still do not know whether it suffices to solve (2).

4 Future Work

Besides proving the above Theorem 3.3, we intend to generalise the notions of RPSs with effects and their solutions in a way that also allows free completely iterative monads (CIM) instead of free monads as used in [MM06]. This might be the right setting for further effects given by the environment monad $MX = X^E$ for a set E or by the non-empty powerset monad $\mathscr{P}^+$. The latter monad should correspond to non-deterministic RPSs in the sense of [AN77]. However, when working with CIMs there seems to be no hope for a general construction of distributive laws of monads as in Lemma 3.1.

Another direction for future work is interpreted solutions of RPSs with effects: we would like to prove that one can obtain unique solutions interpreted in λ-cias/Kleisli cias [MPS09].

References

[AN77] André Arnold and Maurice Nivat. Non deterministic recursive program schemes. In *Fundamentals of computation theory (Proc. Internat. Conf., Poznań-Kórnik, 1977)*, volume 56 of *Lecture Notes in Comput. Sci.*, pages 12–21. Springer, Berlin, 1977.

[Bec69] Jon Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes in Math.*, pages 119–140. Springer Berlin Heidelberg, 1969.

[MM06] Stefan Milius and Lawrence S. Moss. The category theoretic solution of recursive program schemes. *Theoretical Computer Science*, 366:3–59, 2006.

[MPS09] Stefan Milius, Thorsten Palm, and Daniel Schwencke. Complete iterativity for algebras with effects. In *Algebra and Coalgebra in Computer Science (Proc. Third International Conference, Udine, 2009)*, volume 5728 of *Lecture Notes in Comput. Sci.*, pages 34–48. Springer, Berlin, 2009.

[Mul94] Philip S. Mulry. Lifting theorems for kleisli categories. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics—9th international conference*, volume 802 of *LNCS*, pages 304–319. Springer Verlag, 1994.

Strong Relative Monads (Extended Abstract)

Tarmo Uustalu
Institute of Cybernetics at Tallinn University of Technology
Akadeemia tee 21, EE-12618 Tallinn, Estonia
tarmo@cs.ioc.ee

In a paper presented here in Paphos at FoSSaCS [1], we introduced a generalization of monads, called relative monads, to analyze monad-like structures where the underlying functor is not an endofunctor. This study was motivated by several examples of direct relevance for programming theory, such as finite-dimensional vector spaces, the syntaxes of untyped and simply-typed lambda-calculus, but also the arrow types of Hughes [3], as mathematized by Jacobs et al. [4] (in their weak version without the strength). We showed that the Kleisli and Eilenberg-Moore constructions generalize to relative monads and are related to relative adjunctions (with the Kleisli category of an arrow type as a relative monad turning out to be the same as a Freyd category in the sense of Power and Robinson [6]—much as in the work of Jacobs et al.). We also showed that, under reasonable assumptions, relative monads are monoids in the functor category concerned and extend to ordinary monads, giving rise to a coreflection between relative monads and ordinary monads.

But in programming theory applications, we are often interested in strong monads rather than just monads and the standard concept of an arrow type is also that of a strong arrow type. It is therefore natural to ask whether relativization is possible also for strong monads.

In this paper, we study this question and answer it in the affirmative. We introduce the concept of a strong relative monad on a monoidal functor, show that strong arrow types are the same as strong relative monads on the Yoneda embedding (seen as a monoidal functor) and investigate the properties of strong relative monads.

Given a functor J between two categories $\mathbb{J}$ and $\mathbb{C}$, a relative monad on J is given by: (i) an object function $T \in |\mathbb{J}| \to |\mathbb{C}|$, (ii) for any $X \in |\mathbb{J}|$, a map $\eta_X \in \mathbb{C}(JX, TX)$, and (iii) for any $X, Y \in |\mathbb{J}|$ and $k \in \mathbb{C}(JX, TY)$, a map $k^* \in \mathbb{C}(TX, TY)$, satisfying (i) for any $X, Y \in |\mathbb{J}|$, $k \in \mathbb{C}(JX, TY)$, $k = k^* \circ \eta_X$, (ii) for any $X \in |\mathbb{J}|$, $\eta_X^* = \mathsf{id}_{TX}$, and (iii) for any $X, Y, Z \in |\mathbb{J}|$, $k \in \mathbb{C}(JX, TY)$, $\ell \in \mathbb{C}(JY, TZ)$, $(\ell^* \circ k)^* = \ell^* \circ k^*$. Ordinary monads correspond to the special case $\mathbb{J} =_{\mathrm{df}} \mathbb{C}$, $J =_{\mathrm{df}} \mathsf{Id}_{\mathbb{C}}$.

Given a monoidal functor (J, e, m) between two monoidal categories $(\mathbb{J}, I, \otimes)$ and $(\mathbb{C}, I', \otimes')$, we define a strong relative monad on J as a relative monad $(T, \eta, (-)^*)$ on J that also comes with: for any $X, Y \in |\mathbb{J}|$, a map $st_{X,Y} \in \mathbb{C}(TX \otimes' JY, T(X \otimes Y))$, natural in X, Y and making the following diagrams commute:

$$\begin{array}{ccccc} TX \otimes' I' & \xrightarrow{TX \otimes' e} & TX \otimes' JI & \xrightarrow{st_{X,I}} & T(X \otimes I) \\ {\scriptstyle \rho'_X}\downarrow & & & & \downarrow{\scriptstyle \rho_X} \\ TX & = & = & = & TX \end{array} \qquad \begin{array}{ccccc} (TX \otimes' JY) \otimes' JZ & \xrightarrow{st_{X,Y} \otimes' JZ} & T(X \otimes Y) \otimes' JZ & \xrightarrow{st_{X \otimes Y, Z}} & T((X \otimes Y) \otimes Z) \\ {\scriptstyle \alpha'_{TX,TY,TZ}}\downarrow & & & & \downarrow{\scriptstyle T\alpha_{X,Y,Z}} \\ TX \otimes' (JY \otimes' JZ) & \xrightarrow[TX \otimes' m_{Y,Z}]{} & TX \otimes' J(Y \otimes Z) & \xrightarrow[st_{X,Y \otimes Z}]{} & T(X \otimes (Y \otimes Z)) \end{array}$$

$$\begin{array}{ccc} JX \otimes' JY & \xrightarrow{m_{X,Y}} & J(X \otimes Y) \\ {\scriptstyle \eta_X \otimes' JY}\downarrow & & \downarrow{\scriptstyle \eta_{X \otimes Y}} \\ TX \otimes' JY & \xrightarrow[st_{X,Y}]{} & T(X \otimes Y) \end{array} \qquad \begin{array}{ccc} JX \otimes' JY & \xrightarrow{m_{X,Y}} & J(X \otimes Y) \\ {\scriptstyle k \otimes' JY}\downarrow & & \downarrow{\scriptstyle \ell} \\ TX' \otimes' JY & \xrightarrow[st_{X',Y}]{} & T(X' \otimes Y) \end{array} \quad \Rightarrow \quad \begin{array}{ccc} TX \otimes' JY & \xrightarrow{st_{X,Y}} & T(X \otimes Y) \\ {\scriptstyle k^* \otimes' JY}\downarrow & & \downarrow{\scriptstyle \ell^*} \\ TX' \otimes' JY & \xrightarrow[st_{X',Y}]{} & T(X' \otimes Y) \end{array}$$

Arrows on a (small) category $\mathbb{J}$ are the same as relative monads on the Yoneda embedding $\mathbf{Y} \in \mathbb{J} \to [\mathbb{J}^{\mathrm{op}}, \mathbf{Set}]$, defined by $\mathbf{Y}YX =_{\mathrm{df}} \mathbb{J}(X, Y)$. Indeed, an arrow is defined as a functor $R \in \mathbb{J}^{\mathrm{op}} \times \mathbb{J} \to \mathbf{Set}$ (i.e.,

an endoprofunctor on $\mathbb{J}$) together with further data satisfying certain laws, whereas a relative monad on $\mathbf{Y}$ is a functor $T \in \mathbb{J} \to [\mathbb{J}^{\mathrm{op}}, \mathbf{Set}]$, which is the same thing via $R(X,Y) =_{\mathrm{df}} T\,Y\,X$ and $T\,Y\,X =_{\mathrm{df}} R(X,Y)$.

To see that strong arrows on $\mathbb{J}$ are the same as strong relative monads on $\mathbf{Y}$, we first observe that any given monoidal structure $(I, \otimes)$ on $\mathbb{J}$ extends to a monoidal structure $(I', \otimes')$ on $[\mathbb{J}^{\mathrm{op}}, \mathbf{Set}]$ via $I'Z =_{\mathrm{df}} \mathbb{J}(Z,I)$, $(F \otimes' G)Z =_{\mathrm{df}} \int^{X,Y \in |\mathbb{J}|} \mathbb{J}(Z, X \otimes Y) \times (FX \times GY)$ (the Day convolution [2]) whereby $\mathbf{Y}$ becomes a monoidal functor. As $(T\,X \otimes' \mathbf{Y}\,Y)Z = \int^{X',Y' \in |\mathbb{J}|} \mathbb{J}(Z, X' \otimes Y') \times (T\,X\,X' \times \mathbb{J}(Y', Y)) \cong \int^{X' \in |\mathbb{J}|} \mathbb{J}(Z, X' \otimes Y) \times T\,X\,X'$, we have that a strength of a relative monad T on $\mathbf{Y}$ is a natural transformation with components $(st_{X,Y})_Z \in \int^{X' \in |\mathbb{J}|} \mathbb{J}(Z, X' \otimes Y) \times T\,X\,X' \to T(X \otimes Y)Z$. The strength of an arrow R is a (di)natural transformation with components $fst_{X',X,Y} \in R(X',X) \to R(X' \otimes Y, X \otimes Y)$, which amounts to the same.

It is viable that the view of strong arrows as strong relative monads on the Yoneda embedding may provide considerations about good ways for defining an arrow-based metalanguage. We would like to check whether these lead to the design of Lindley et al. [5].

Acknowledgement This research is being supported by the Estonian Science Foundation grant no. 6940.

References

[1] T. Altenkirch, J. Chapman, T. Uustalu. Monads need not be endofunctors. In C.-H. L. Ong, ed., *Proc. of 13th Int. Conf. on Foundations of Software Science and Computation Structures, FoSSaCS 2010 (Paphos, March 2010)*, v. 6014 of *Lect. Notes in Comput. Sci.*, pp. 297–311. Springer, 2010.

[2] B. Day. On closed categories of functors. In *Reports of the Midwest Category Theory Seminar IV*, v. 137 of *Lect. Notes in Math.*, pp. 1–38. Springer, 1970.

[3] J. Hughes. Generalising monads to arrows. *Sci. of Comput. Program.*, v. 37, n. 1–3, pp. 67–111, 2000.

[4] B. Jacobs, C. Heunen, I. Hasuo. Categorical semantics for arrows. *J. of Funct. Program.*, v. 19, n. 3–4, pp. 403–438, 2009.

[5] S. Lindley, P. Walder, J. Yallop. The arrow calculus. *J. of Funct. Program.*, v. 20, n. 1, pp. 51–69, 2010.

[6] J. Power, E. Robinson. Premonoidal categories and notions of computation. *Math. Struct. in Comput. Sci.*, v. 7, n. 5, pp. 453–468, 1997.