Processes Parametrised by an Algebraic Theory

Todd Schmid
Department of Computer Science, University College London, UK

Wojciech Rozowski
Department of Computer Science, University College London, UK

Alexandra Silva
Department of Computer Science, Cornell University, Ithaca, NY, USA

Jurriaan Rot
Institute for Computing and Information Sciences, Radboud University, Nijmegen, The Netherlands

Abstract

We develop a (co)algebraic framework to study a family of process calculi with monadic branching structures and recursion operators. Our framework features a uniform semantics of process terms and a complete axiomatisation of semantic equivalence. We show that there are uniformly defined fragments of our calculi that capture well-known examples from the literature like regular expressions modulo bisimilarity and guarded Kleene algebra with tests. We also derive new calculi for probabilistic and convex processes with an analogue of Kleene star.

2012 ACM Subject Classification Theory of computation → Program reasoning

Keywords and phrases process algebra, program semantics, coalgebra, regular expressions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.132

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Related Version *Full Version*: `https://arxiv.org/abs/2202.06901`

Funding *Todd Schmid*: Partially supported by ERC grant Autoprobe (grant agreement 101002697). *Wojciech Rozowski*: Partially supported by ERC grant Autoprobe (grant agreement 101002697). *Alexandra Silva*: Partially supported by ERC grant Autoprobe (grant agreement 101002697) and a Royal society Wolfson fellowship.

1 Introduction

The theory of processes has a long tradition, notably in the study of concurrency, pioneered by seminal works of Milner [30, 29] and many others [2]. In labelled transition systems, a popular model of computation in process theory, processes branch nondeterministically. This means that any given action or observation transitions a starting state into any member of a predetermined set of states. In Milner's CCS [29], nondeterminism appears as a binary operation that constructs from a pair of programs e and f the program $e + f$ that nondeterministically chooses between executing either e or f. This acts precisely like the join operation in a semilattice. In fact, elements of a free semilattice are exactly sets, as the free semilattice generated by a collection X is the set $\mathcal{P}^+_\omega X$ of finite nonempty subsets of X [26]. This is our first example of a more general phenomenon: the type of branching in models of process calculi can often be captured with an algebraic theory.

A second example appears in the probabilistic process algebra literature, where the process denoted $e +_p f$ flips a weighted coin and runs e with probability p and f with probability $1 - p$. The properties of $+_p$ are axiomatised and studied in convex algebra, an often revisited algebraic theory of probability [1, 47, 35, 46]. The free convex algebra on a set X is the set $\mathcal{D}_\omega X$ of finitely supported probability distributions on X [46, 11, 23].


A third example is guarded Kleene algebra with tests (GKAT), where the process $e +_b f$ proceeds with e if a certain Boolean predicate b holds and otherwise proceeds with f, emulating the `if-then-else` constructs of imperative programming languages [27, 8, 28]. If the predicates are taken from a finite Boolean algebra $2^{\mathtt{At}}$, the free algebra of `if-then-else` clauses on a set X is the function space $X^{\mathtt{At}}$. This explains why adjacency sets for tree models of GKAT programs take the form of functions $\mathtt{At} \to X$.

This paper proposes a framework in which these languages can be uniformly described and studied. We use the *algebra of regular behaviours* (or ARB) introduced in [30] as a prototypical example. ARB employs nondeterministic choice as a branching operation, prefixing of terms by atomic actions, a constant representing deadlock, variables, and a recursion operator for each variable. Specifications are interpreted using structural operational semantics in the style of [34], which sees the set `Exp` of all process terms as one large labelled transition system. This is captured succinctly as a *coalgebra*, in this case a function

$$\beta : \mathtt{Exp} \to \mathcal{P}(V + A \times \mathtt{Exp}) \tag{1}$$

Only finitely branching processes can be specified in ARB, so we will replace $\mathcal{P}$ with $\mathcal{P}_\omega$ in (1). From a technical point of view, $\mathcal{P}_\omega$ is the monad on the category **Sets** of sets and functions presented by the algebraic theory of semilattices with bottom.

By substituting the finite powerset functor in (1) with other monads presented by algebraic theories, we obtain a parametrised family of process types that covers the examples above and a general framework for studying the processes of each type. Instantiating the framework with an algebraic theory gives a fully expressive specification language for processes and a complete axiomatisation of behavioural equivalence for specifications.

One striking feature of many of the specification languages we construct is that they contain a fragment consisting of nonstandard analogues of regular expressions. We call these expressions *star expressions* and the fragment composed of star expressions the *star fragment*. Star fragments extend several existing analogues of basic regular algebra found in the process theory literature, including basic process algebra [5] and Andova's probabilistic basic process algebra [1], by adding recursion operators modelled after the Kleene star.

Milner is the first to notice the star fragment of ARB in [30]. He observes that the algebra of processes denoted by star expressions is more unruly than Kleene's algebra of regular languages, and that it is not clear what the appropriate axiomatisation should be. He offers a reasonable candidate based on Salomaa's first axiomatisation of Kleene algebra [39], but ultimately leaves completeness as an open problem. This problem has been subjected to many years of extensive research [15, 14, 16, 3, 20, 19]. A potential solution has recently been announced by Clemens Grabmayer and will appear in the upcoming LICS.

Replacing nondeterministic choice with the `if-then-else` branching structure of GKAT, we obtain the process behaviours explored in the recent rethinking of the language [40]. This makes the open problem of axiomatising GKAT (without the use of extremely powerful axioms like the *Uniqueness Axiom* of [44]), stated first in [44] and again in [40], yet another problem of axiomatising an algebra of star expressions. Our general characterisation of star expressions puts all these languages under one umbrella, and shows how they are derived canonically from a single abstract framework.

In summary, the contributions of this paper are as follows:

- We present a family of process types parametrised by an algebraic theory (Section 2) together with a uniform syntax and operational semantics (Section 3). We show how these can be instantiated to concrete algebraic theories, including guarded semilattices and pointed convex algebras. These provide, respectively, a calculus of processes capturing control flow of simple imperative programs and a calculus of probabilistic processes.

- We define an associated denotational semantics and show that it agrees with the operational semantics (Section 4). This coincidence result is important in order to prove completeness of the uniform axiomatisation we propose for each process type (Section 5).
- Finally, we study the star fragment of our parameterised family and propose a sound axiomatisation for this fragment (Section 6). We show that star fragments of concrete instances of our calculi yield known examples in the literature, e.g. Guarded Kleene Algebra with tests (GKAT) [44, 40] and probabilistic processes of Stark and Smolka [45].

Related work is surveyed in Section 7, and future research directions are discussed in Section 8.

2 A Parametrised Family of Process Types

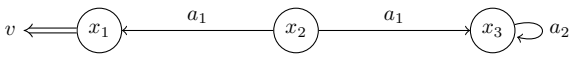
In this section, we present a family of process types parametrised by a certain kind of algebraic theory. The processes we care about are stateful, meaning they consist of a set of states and a suitably structured set of transitions between states. Stateful systems fit neatly into the general framework of *universal coalgebra* [37], which stipulates that the type of structure carried by the transitions can be encoded in an endofunctor on the category **Sets** of sets and functions. Formally, given a functor $B : \mathbf{Sets} \to \mathbf{Sets}$, a *$B$-coalgebra* is a pair (X, β) consisting of a set X of *states* and a *structure* map $\beta : X \to BX$. A *coalgebra homomorphism* $h : (X, \beta) \to (Y, \vartheta)$ is a function $h : X \to Y$ satisfying $\vartheta \circ h = B(h) \circ \beta$. Many types of processes found in the literature coincide with B-coalgebras for some B, and so do their homomorphisms. For example, finitely branching labelled transition systems are $\mathcal{P}_\omega(A \times \mathrm{Id})$-coalgebras, and deterministic Moore automata are $O \times \mathrm{Id}^A$-coalgebras [36].

In this paper, we consider coalgebras for functors of the form

$$B_M := M(V + A \times \mathrm{Id}) \tag{2}$$

for fixed sets V and A and a specific kind of functor $M : \mathbf{Sets} \to \mathbf{Sets}$. Intuitively, there are two layers to the process behaviours we care about: one layer consists of either an *output variable* in V or an *action* from A that moves on to another state, and the other layer (encoded by M) combines output variables and action steps in a structured way.

▶ **Example 1.** When $M = \mathcal{P}_\omega$, we obtain Milner's nondeterministic processes [30]. Coalgebras for $B_{\mathcal{P}_\omega}$ are functions of the form $\beta : X \to \mathcal{P}_\omega(V + A \times X)$, or labelled transition systems with an additional decoration by variables. Write $x \xrightarrow{a} y$ to mean $(a, y) \in \beta(x)$ and $x \Rightarrow v$ to mean $v \in \beta(x)$. The image below posits a well-defined $B_{\mathcal{P}_\omega}$-coalgebra



Its state space is $\{x_1, x_2, x_2\}$, A includes a_1 and a_2, and v is a variable in V.

Algebraic Theories and Their Monads

We are particularly interested in B_M-coalgebras when M is the functor component of a monad (M, η, μ) that is presented by an algebraic theory capturing a type of branching. A monad consists of natural transformations $\eta : \mathrm{Id} \Rightarrow M$ and $\mu : MM \Rightarrow M$, called the *unit* and *multiplication* respectively, satisfying two laws: $\mu \circ \eta_M = \mathrm{id}_M = \mu \circ M(\eta)$ and $\mu_M \circ \mu = M(\mu) \circ \mu$. For our purposes, an *algebraic theory* is a pair (S, E) consisting of a polynomial endofunctor $S = \coprod_{\sigma \in I} \mathrm{Id}^{n_\sigma}$ on **Sets** called an *algebraic signature* and a set E of equations in the signature S. An element σ of I should be thought of as an operation

with arity n_σ. An algebraic theory (S, E) *presents* a monad (M, η, μ) if there is a natural transformation $\rho : SM \Rightarrow M$ such that for any set X, (MX, ρ_X) is the free (S, E)-algebra on X. That is, (MX, ρ_X) satisfies E and for any S-algebra (Y, φ) also satisfying E and any function $h : X \to Y$, there is a unique S-algebra homomorphism $\hat{h} : (MX, \rho_X) \to (Y, \varphi)$ such that $h = \hat{h} \circ \eta$. This universal property implies that any two presentations of a given algebraic theory are isomorphic, so we speak simply of "the" monad presented by an algebraic theory.

▶ **Example 2.** The finite powerset functor is part of the monad $(\mathcal{P}_\omega, \{-\}, \bigcup)$ that is presented by the theory of semilattices (with bottom). The theory of semilattices is the pair $(1+\mathrm{Id}^2, \mathsf{SL})$, since the arity of a constant operation is 0 and $+$ is a binary operation, and SL consists of

$$x + 0 \overset{(\mathsf{SL1})}{=} x \qquad x + x \overset{(\mathsf{SL2})}{=} x \qquad x + y \overset{(\mathsf{SL3})}{=} y + x \qquad x + (y + z) \overset{(\mathsf{SL4})}{=} (x + y) + z$$

Not every algebraic theory has such a familiar presentation as the theory of semilattices, but it is nevertheless true that every algebraic theory presents a monad. If we let S^*X denote the set of S-*terms*, expressions built from X and the operations in S, then (S, E) automatically presents the monad (M, η, μ) where $MX = (S^*X)/\mathsf{E} := \{[q]_\mathsf{E} \mid q \in S^*X\}$ is the set of E-congruence classes of S-terms, η computes congruence classes of variables, and μ evaluates terms. This is witnessed by letting the transformation ρ be the restriction of μ to the operations of S on S-terms. We take this to be the default presentation of an arbitrary algebraic theory.

Our aim is to develop a (co)algebraic framework for studying B_M-coalgebras when M is the functor part of a monad presented by an algebraic theory. We will make three assumptions about the algebraic theories. First, we rule out the case of M being the constant 1 functor.

▶ **Assumption 1.** *The theory* E *is* nontrivial*, meaning that the equation* $x = y$ *is not a consequence of* E *for distinct* x *and* y*.*

This is equivalent to requiring that the unit η is injective. That is, the E-congruence classes $[x]_\mathsf{E}$ and $[y]_\mathsf{E}$ in MX are distinct for distinct variables x and y in X.

Second, we assume the existence of a constant symbol denoting deadlock, which might occur when recursing on unguarded programs.

▶ **Assumption 2.** *Algebraic theories contain a designated constant* 0*.*

Finally, to keep the specifications of processes finite, we make the following assumption despite the fact that it has no bearing on the results presented before Section 5.

▶ **Assumption 3.** *Each operation from* S *has a finite arity.*

We conclude this section with examples of algebraic theories and the monads they present.

▶ **Example 3.** For a fixed finite set $\mathtt{At}$ of *atomic tests*, the algebraic theory of *guarded semilattices* is the pair $(1 + \coprod_{b \subseteq \mathtt{At}} \mathrm{Id}^2, \mathsf{GS})$, where GS consists of the equations

$$x +_b x \overset{(\mathsf{GS1})}{=} x \qquad x +_\mathtt{At} y \overset{(\mathsf{GS2})}{=} x \qquad x +_b y \overset{(\mathsf{GS3})}{=} y +_{\bar{b}} x \qquad (x +_b y) +_c z \overset{(\mathsf{GS4})}{=} x +_{bc} (y +_c z)$$

Here, $+_b$ is the binary operation associated with the subset $b \subseteq \mathtt{At}$, $\bar{b} := \mathtt{At} \setminus b$, and $bc := b \cap c$. The theory of guarded semilattices is presented by the monad $((1 + \mathrm{Id})^\mathtt{At}, \lambda\xi.(-), \Delta^*)$, where $(\lambda\xi.x)(\xi) = x$ and $\Delta^*(F)(\xi) = F(\xi)(\xi)$. The idea is that $+_b$ acts like an `if-then-else` clause in an imperative program. This is reflected in a free guarded semilattice $((1 + X)^\mathtt{At}, \rho_X)$, where for a pair of maps $h_1, h_2 : \mathtt{At} \to X$ we define

$$\rho_X(h_1 +_b h_2)(\xi) := \begin{cases} h_1(\xi) & \text{if } \xi \in b \\ h_2(\xi) & \text{otherwise} \end{cases}$$

The theory of guarded semilattices dates back to the algebras of `if-then-else` clauses studied in [28, 27, 8, 7]. For instance, guarded semilattices are examples of *McCarthy algebras*, introduced by Manes in [27].[1]

▶ **Example 4.** The theory of *pointed convex algebras* studied in [12] is $(1 + \coprod_{p\in[0,1]} \mathrm{Id}^2, \mathsf{CA})$, where CA consists of the equations

$$x +_p x \overset{(\mathsf{CA1})}{=} x \qquad x +_1 y \overset{(\mathsf{CA2})}{=} x \qquad x +_p y \overset{(\mathsf{CA3})}{=} y +_{\bar{p}} x \qquad (x +_p y) +_q z \overset{(\mathsf{CA4})}{=} x +_{pq} (y +_{\frac{q\bar{p}}{1-pq}} z)$$

Here, $+_p$ is the binary operation with index $p \in [0,1]$, $\bar{p} := 1 - p$, and $pq \neq 1$. This theory presents the pointed finite subprobability distribution monad $(\mathcal{D}_\omega(1 + \mathrm{Id}), \delta_{(-)}, \sum)$, where

$$\mathcal{D}_\omega(1 + X) = \left\{ \theta : X \to [0,1] \;\middle|\; \begin{array}{c} \{x \mid \theta(x) > 0\} \text{ is finite} \\ \sum_{x\in X} \theta(x) \leq 1 \end{array} \right\}$$

for any set X, and for any $x \in X$, $\theta \in \mathcal{D}_\omega(1 + X)$, and $\Theta \in \mathcal{D}_\omega(1 + \mathcal{D}_\omega(1 + X))$,

$$\delta_x(y) = [x = y?] \qquad \sum(\Theta)(\theta) = \sum_{y\in X} \Theta(\theta) \cdot \theta(y)$$

This is witnessed by the transformation ρ that takes 0 to the trivial subdistribution and computes the Minkowski sum $\rho_X(\theta +_p \psi) = p\cdot\theta + (1-p)\cdot\psi$ for each $p \in [0,1]$, $\theta, \psi \in \mathcal{D}_\omega(1+X)$.

▶ **Example 5.** The theory of *pointed convex semilattices* studied in [12, 49, 10] combines the theory of semilattices and the theory of convex algebras. It has both a binary operation $+$ mimicking nondeterministic choice and the probabilistic choice operations $+_p$ indexed by $p \in [0,1]$. Formally, it is given by the pair $(1 + \mathrm{Id}^2 + \coprod_{p\in[0,1]} \mathrm{Id}^2, \mathsf{CS})$, where CS is the union of SL, CA, and the distributive law

$$(x + y) +_p z \overset{(\mathsf{D})}{=} (x +_p z) + (y +_p z)$$

This theory presents the pointed convex powerset monad $(\mathcal{C}, \eta^{\mathcal{C}}, \mu^{\mathcal{C}})$, where $\mathcal{C}X$ is the set of finitely generated convex subsets of $\mathcal{D}_\omega(1 + X)$ containing δ_0, and for $x \in X$ and $Q \in \mathcal{C}\mathcal{C}X$,

$$\eta^{\mathcal{C}}(x) = \{p\cdot\delta_x \mid p \in [0,1]\} \qquad \mu^{\mathcal{C}}(Q) = \bigcup_{\Theta\in Q} \left\{ \sum_{U\in\mathcal{C}_0 X} \Theta(U) \cdot \theta_U \;\middle|\; (\forall U \in \mathcal{C}_0 X)\ \theta_U \in U \right\}$$

The witnessing transformation $\rho^{\mathcal{C}}$ takes 0 to $\{\delta_0\}$, computes the Minkowski sum (extended to subsets) in place of $+_p$, and interprets the $+$ operation as the convex union

$$\rho^{\mathcal{C}}_X(U + V) = \{p\cdot\theta_1 + (1-p)\cdot\theta_2 \mid p \in [0,1], \theta_1 \in U, \theta_2 \in V\}$$

3 Specifications of Processes

Fix an algebraic theory (S, E) presenting a monad (M, η, μ). In this section, we give a syntactic and uniformly defined specification system for B_M-coalgebras and an associated operational semantics. We are primarily concerned with the specifications of finite processes, and indeed the process terms we construct below denote processes with finitely many states. The converse is also true, that every finite B_M-coalgebra admits a specification in the form of a process term, but we defer this result to Section 5 because of its relevance to the completeness theorem there.

[1] More information on the theory of guarded semilattices can be found in [42, Appendix A].

$$\epsilon(v) = \eta(v) \qquad \epsilon(\sigma(e_1, \ldots, e_n)) = \sigma(\epsilon(e_1), \ldots, \epsilon(e_n))$$
$$\epsilon(ae) = \eta((a, e)) \qquad \epsilon(\mu v\ e) = \epsilon(e)[\mu v\ e // v]$$

Figure 1 Operational semantics of process terms. Here, $v \in V$, $a \in A$, and $e, e_i \in \mathtt{Exp}$.

The syntax of our specifications consists of variables from an infinite set V, actions from a set A, and operations from S. The set $\mathtt{Exp}$ of *process terms* is given with the grammar

$$e, e_i ::= 0 \mid v \mid \sigma(e_1, \ldots, e_n) \mid ae \mid \mu v\ e$$

where $v \in V$, $a \in A$, and σ is an S-operation. Abstractly, process terms form the initial Σ_M-algebra $(\mathtt{Exp}, \alpha)$, where $\Sigma_M : \mathbf{Sets} \to \mathbf{Sets}$ is the functor defined by

$$\Sigma_M := S + V + A \times \mathrm{Id} + V \times \mathrm{Id}$$

and the algebra map $\alpha : \Sigma_M \mathtt{Exp} \to \mathtt{Exp}$ evaluates Σ_M-terms.

Intuitively, the symbol 0 is the designated constant of S denoting the *deadlock* process, which takes no action. Output variables are used in one of two ways, depending on the expression in which they appear. A variable v is *free* in an expression e if it does not appear within the scope of μv and *bound* otherwise. If v is free in e, then v denotes "output v". Otherwise, v denotes a *goto* statement that returns the computation to the μv that binds v. The process $\sigma(e_1, \ldots, e_n)$ is the process that branches into $e_1, \ldots, e_n$ using an n-ary operation σ as the branching constructor. The expression ae denotes the process that performs the action a and then proceeds with e. Finally, $\mu v\ e$ denotes recursion in the variable v.

Small-step Semantics

Next we give a small-step (operational) semantics to process terms that is uniformly defined for the process types in our parametrised family. Many of the algebraic theories we consider lack a familiar presentation, which ultimately prevents the corresponding semantics from taking the traditional form of a set of inference rules describing transition relations. We take an abstract approach instead by defining a B_M-coalgebra structure $\epsilon : \mathtt{Exp} \to B_M \mathtt{Exp}$ that mirrors the intuitive descriptions of the executions of process terms above. The formal description of ϵ is summarised in Figure 1.

The operational interpretation of the recursion operators requires further explanation. Intuitively, $\mu v\ e$ performs the process denoted by e until it reaches an exit in channel v, at which point it loops back to the beginning. However, this is really only an accurate description of recursion in v when e performs an action before exiting in v. For example, the process $\mu v\ v$ not only never exits in channel v, but it also never performs any action at all. Thus, the operational interpretation of $\mu v\ v$ is indistinguishable from that of deadlock. We deal with this issue as follows: if an exit in channel v is immediately reached by a branch of e, then we replace that exit with deadlock in $\mu v\ e$. Formally, we say that a variable v is *guarded* in a process term e if (i) $e \in V \setminus \{v\}$, (ii) $e = af$ or (iii) $e = \mu v\ f$ for some $f \in \mathtt{Exp}$, or (iv) either $e = \mu u\ e_1$ or (v) $e = \sigma(e_1, \ldots, e_n)$ and v is guarded in e_i for each $i \leq n$. In our calculus, we syntactically allow for recursion in unguarded variables, but one should keep in mind that those variables are ultimately deadlock under the recursion operator.

The operational interpretation of recursion is formally defined using a *guarded syntactic substitution operator* $[g//v] : B_M\mathtt{Exp} \to B_M\mathtt{Exp}$,[2] a variant of the usual syntactic substitution of variables. Given $g \in \mathtt{Exp}$, we first define $[g//v]$ by induction on $S^*(V + A \times \mathtt{Exp})$ as

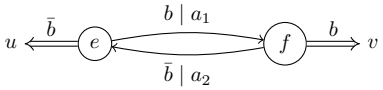
$$u[g//v] = \begin{cases} \eta(u) & u \neq v \\ \eta(0) & u = v \end{cases} \qquad \begin{aligned} (a, f)[g//v] &= (a, f[g/v]) \\ \sigma(p_1, \ldots, p_n)[g//v] &= \sigma(p_1[g//v], \ldots, p_n[g//v]) \end{aligned}$$

where $u \in V$, $p_i \in S^*(V + A \times \mathtt{Exp})$, $f \in \mathtt{Exp}$, and $[g/v]$ replaces free occurrence of v with g. The following lemma completes the description of the operational semantics of process terms.

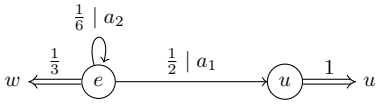
▶ **Lemma 6.** *For any $g \in \mathtt{Exp}$ and $v \in V$, the map $[g//v]$ factors uniquely through $B_M\mathtt{Exp}$.*

Formally, the map ϵ assigns to each process term e an E-congruence class $\epsilon(e)$ of terms from $S^*(V + A \times \mathtt{Exp})$. A term from $S^*(V + A \times \mathtt{Exp})$ is a combination of variables v and transition-like pairs (a, e_i), so there is often only a small conceptual leap from the coalgebra structure ϵ to a more traditional representation of transitions as decorated arrows. We provide the following examples as illustrations of this phenomenon, as well as the specification languages and operational semantics of terms defined above.[3]

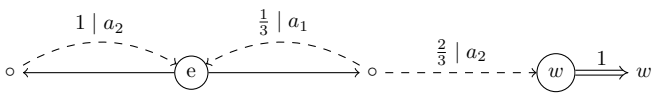
▶ **Example 7.** The *algebra of control flows*, or ACF, is obtained from the theory of guarded semilattices of Example 3 and $M = (1 + Id)^{\mathtt{At}}$. Given a structure map $\beta : X \to B_{(1+\mathrm{Id})^{\mathtt{At}}}X$ and $b \subseteq \mathtt{At}$, write $x \xrightarrow{b|a} y$ if $\beta(x)(\xi) = (a, y)$ for all $\xi \in b$, and $x \overset{b}{\Longrightarrow} v$ if $\beta(x)(\xi) = v$ for all $\xi \in b$. The operational semantics returns the constant map $\lambda\xi.v$ given a variable $v \in V$ and interprets conditional choice as guarded union. For example, let $e = \mu w\ (a_1(v +_b a_2 w) +_b u)$ and $f = v +_b a_2 e$. The process denoted by e is



▶ **Example 8.** The *algebra of probabilistic actions*, or APA, is obtained from the theory of pointed convex algebras of Example 4 and $M = \mathcal{D}_\omega(1 + Id)$. For a structure map $\beta : X \to B_{\mathcal{D}_\omega(1+\mathrm{Id})}X$, write $x \xrightarrow{k|a} y$ when $\beta(x)(a, y) = k$ and $e \overset{k}{\Longrightarrow} v$ when $\beta(e)(v) = k$. The operational semantics returns the Dirac distribution δ_v for $v \in V$ and interprets probabilistic choice as the Minkowski sum. The process denoted by $e = \mu v\ (a_1 u +_{\frac{1}{2}} (a_2 v +_{\frac{1}{3}} w))$ is



▶ **Example 9.** The *algebra of nondeterministic probabilistic actions*, or ANP, is obtained from the theory of pointed convex semilattices of Example 5. For a structure map $\beta : X \to B_{\mathcal{C}}X$, write $x \to\!\circ\xrightarrow{k|a} y$ to mean there is a $\theta \in \beta(x)$ such that $\theta(a, y) = k$, and $x \overset{k}{\Longrightarrow} v$ to mean there is a $\theta \in \beta(x)$ with $\theta(v) = k$. The operational semantics returns $\eta^{\mathcal{C}}(v)$ given $v \in V$, interprets nondeterministic choice as convex union, and replaces probabilistic choice with Minkowski sum. For example, $e = \mu v\ ((a_1 v +_{\frac{1}{3}} a_2 w) + a_2 v)$ denotes



[2] Technically, it is only partially defined. See [42, Appendix C] for details.
[3] See [42, Appendix B] in the full version of the paper.

$$\zeta(\gamma(v)) = [v]_{\mathsf{E}} \qquad \zeta(\gamma(\sigma(t_1,\ldots,t_n))) = [\sigma(\zeta(t_1),\ldots,\zeta(t_n))]_{\mathsf{E}}$$
$$\zeta(\gamma(a,t)) = [(a,t)]_{\mathsf{E}} \qquad \zeta(\gamma(\mu v\ t)) = \zeta(t)\{\gamma(\mu v\ t)//v\}$$

Figure 2 The Σ_M-algebra structure of (Z,γ). Here, $v \in V$, $a \in A$, $t, t_i \in Z$ for $i \leq n$, and σ is an n-ary operation from S. By Lambek's lemma [25], $\zeta : Z \to B_M Z$ is a bijection, so the first three equations determine $\gamma : V + SZ + A \times V \to Z$. The fourth is a behavioural differential equation [36].

4 Behavioural Equivalence and the Final Coalgebra

In this section, we relate the operational semantics arising from the coalgebra structure on $\mathtt{Exp}$ in the previous section to a denotational semantics, which arises through the definition of a suitable algebra structure on the domain of process behaviours.

For an arbitrary functor $B : \mathbf{Sets} \to \mathbf{Sets}$, a *behaviour* is a state of the *final* B-coalgebra (Z,ζ), the unique (up to isomorphism) coalgebra (if it exists) such that there is exactly one homomorphism $!_\beta : (X,\beta) \to (Z,\zeta)$ from every B-coalgebra (X,β). It follows from general considerations that the functor B_M admits a final coalgebra [37]. The universal property of the final B_M-coalgebra produces the homomorphism $!_\epsilon : (\mathtt{Exp},\epsilon) \to (Z,\zeta)$. The behaviour $!_\epsilon(e)$ is called the *final (coalgebra) semantics* of e, also known as its operational semantics [38].

For example, the final $B_{\mathcal{P}_\omega}$-coalgebra consists of bisimulation equivalence classes of finite and infinite labelled trees of a certain form [4]. In this setting, $(\mathtt{Exp},\epsilon)$ is a labelled transition system and the final semantics $!_\epsilon$ constructs a tree from a process term by unrolling. Intuitively, this captures the behaviour of a specification by encoding all possible actions and outgoing messages at each time-step in its execution.

In addition to forming the state space of the final B_M-coalgebra, the set of process behaviours also carries the structure of a Σ_M-algebra (Z,γ), summarised in Figure 2. Now, $(\mathtt{Exp},\alpha)$ is the *initial* Σ_M-algebra, which in particular means there is a unique algebra homomorphism $[\![-]\!] : (\mathtt{Exp},\alpha) \to (Z,\gamma)$. The behaviour $[\![e]\!]$ is called the *initial (algebra) semantics* of e [18], and provides a denotational semantics to our process calculus.

The algebra structure $\gamma : \Sigma_M Z \to Z$ of (Z,γ) can be seen as a reinterpretation of the programming constructs of the language $\mathtt{Exp}$ that mimics the operational semantics of process terms. The basic constructs are the content of the first three equations in Figure 2: output variables are evaluated so as to behave like the variables of $(\mathtt{Exp},\epsilon)$, the behaviour at performs a and moves on to t, and $\sigma(t_1,\ldots,t_n)$ branches into the behaviours $t_1,\ldots,t_n$ with additional structure determined by the operation σ. Interpreting recursive behaviours like $\mu v\ t$ requires coalgebraic analogues of syntactic and guarded syntactic substitution from Section 3.

For a given behaviour $s \in Z$ and a variable $v \in V$, the *behavioural substitution* of s for v is the map $\{s/v\} : Z \to Z$ defined by the identity

$$\zeta(t\{s/v\}) = \begin{cases} \zeta(s) & \zeta(t) = [v]_{\mathsf{E}} \\ [u]_{\mathsf{E}} & \zeta(t) = [u]_{\mathsf{E}} \neq [v]_{\mathsf{E}} \\ [(a, r\{s/v\})]_{\mathsf{E}} & \zeta(t) = [(a,r)]_{\mathsf{E}} \\ \sigma(\zeta(t_1\{s/v\}),\ldots,\zeta(t_n\{s/v\})) & \zeta(t) = [\sigma(\zeta(t_1),\ldots,\zeta(t_n))]_{\mathsf{E}} \end{cases}$$

for any $t \in Z$. The *guarded behavioural substitution* of s for v is constructed in analogy with guarded syntactic substitution from the previous section. We start by defining guarded behavioural substitution in $S^*(V + A \times Z)$ as

$$u\{s//v\} = \begin{cases} u & u \neq v \\ 0 & u = v \end{cases} \qquad \begin{aligned} (a,r)\{s//v\} &= (a, r\{s/v\}) \\ \sigma(r_1,\ldots,r_n)\{s//v\} &= \sigma(r_1\{s//v\},\ldots,r_n\{s//v\}) \end{aligned}$$

where $u \in V$, $a \in A$, and $r, r_i \in Z$ for $i \leq n$. This map lifts to an operator $B_M Z \to B_M Z$ for the same reason as the guarded syntactic substitution operator. This completes the description of the algebraic structure of (Z, γ) in Figure 2.

▶ **Theorem 10.** *Let* $[\![-]\!]$ *be the unique algebra homorphism* $(\mathtt{Exp}, \alpha) \to (Z, \gamma)$. *For any process term* $e \in \mathtt{Exp}$, *we have* $!_\epsilon(e) = [\![e]\!]$.

In other words, the final semantics given with respect to operational rules in Exp coincides with the initial semantics given with respect to the programming constructs in Z. Consequently, we write $[\![-]\!]$ in place of $!_\epsilon$ and simply refer to $[\![e]\!]$ as the semantics of e.

5 An Axiomatisation of Behavioural Equivalence

An important corollary of Theorem 10 is that behavioural equivalence is a Σ_M-congruence on $(\mathtt{Exp}, \alpha)$, meaning that it is preserved by all the program constructs of Σ_M. This opens the door to the possibility of deriving behavioural equivalences between process terms from just a few axioms. The purpose of this section is to show that all behavioural equivalences between process terms can be derived from the equations in E presenting (M, η, μ) as well as three axiom schemas concerning the recursion operators.

The first two out of the three recursion axiom schemas are

$$\text{(R1)} \quad \mu v\ e = e[\mu v\ e//v] \qquad\qquad \text{(R2)} \quad \frac{w \text{ not free in } e}{\mu v\ e = \mu w\ (e[w/v])}$$

Above, $e[\mu v\ e//v]$ is the expression obtained by replacing every guarded free occurrence of v in e with the expression $\mu v\ e$ and every unguarded occurrence of v in e with 0, in analogy with the operator on $B_M\mathtt{Exp}$ of the same name.[4]

The axiom (R1) essentially allows for a sort of guarded unravelling of recursive terms. This has the effect of identifying $\mu v\ v$ with 0, for example, as well as $\mu v\ av$ with $a(\mu v\ av)$. The latter satisfies our intuition that $\mu v\ av$ should solve the recursive specification $x = ax$ in the indeterminate x. The axiom (R2) allows for recursion variables to be swapped for fresh variables. This amounts to the observation that pairs of terms like $\mu v\ av$ and $\mu w\ aw$ should both denote the unique solution to $x = ax$.

The third recursion axiom schema can be stated in the form of the proof rule

$$\text{(R3)} \quad \frac{g = e[g/v] \quad v \text{ guarded in } e}{g = \mu v\ e}$$

We let R denote the set of equations derived from (R1)-(R3), and we let $\equiv$ denote the smallest congruence in $(\mathtt{Exp}, \alpha)$ containing the set of equations derived from E and R. When we refer to examples like ARB, ACF, APA, and ANP, we are often identifying each of these with their associated algebras $(\mathtt{Exp}/{\equiv}, \hat{\alpha})$ of process terms modulo $\equiv$.

[4] Indeed, the identity $\epsilon(e[\mu v\ e//v]) = \epsilon(e)[\mu v\ e//v]$ holds for all $e \in \mathtt{Exp}$ and $v \in V$ [42, Lemma C.9].

Soundness

We would like to argue that $\equiv$ is *sound* with respect to behavioural equivalence, meaning that $[\![e]\!] = [\![f]\!]$ whenever $e \equiv f$. This is indeed the case, and can be derived from the fact that the set of congruence classes of process terms itself forms a B_M-coalgebra. For an arbitrary function $h : X \to Y$, call the set $\ker(h) := \{(x, y) \mid h(x) = h(y)\}$ the *kernel* of h.

▶ **Lemma 11.** *The congruence $\equiv$ is the kernel of a coalgebra homomorphism.*

We write $[-]_\equiv : \mathtt{Exp} \to \mathtt{Exp}/{\equiv}$ for the quotient map and $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$ for the coalgebra structure on $\mathtt{Exp}/{\equiv}$ making $[-]_\equiv$ a coalgebra homomorphism (there is at most one such coalgebra structure [37]). As $[\![-]\!]$ is the unique coalgebra homomorphism $(\mathtt{Exp}, \epsilon) \to (Z, \zeta)$, and because there is also a coalgebra homomorphism $!_{\bar{\epsilon}} : (\mathtt{Exp}/{\equiv}, \bar{\epsilon}) \to (Z, \zeta)$, it must be the case that $!_{\bar{\epsilon}} \circ [-]_\equiv = [\![-]\!]$. By Lemma 11, if $e \equiv f$, then $[\![e]\!] = !_{\bar{\epsilon}}([e]_\equiv) = !_{\bar{\epsilon}}([f]_\equiv) = [\![f]\!]$. This establishes the following.

▶ **Theorem 12** (Soundness). *Let $e, f \in \mathtt{Exp}$. If $e \equiv f$, then $[\![e]\!] = [\![f]\!]$.*

Soundness allows us to derive at least a subset of all the behavioural equivalences between process terms from the axioms in E and R. If our aspiration were simply to have a set of behaviour-preserving code-transformations, then we could simply stop here and be satisfied, since in principle we could see the axioms of E and R as rewrite rules that satisfy this purpose.

Completeness

Aiming a bit higher than deriving only a subset of the behavioural equivalences between process terms, we move on to show the converse of Theorem 12, that $\equiv$ is *complete* with respect to behavioural equivalence. We use [41, Lemma 5.1], which can be stated as follows.

▶ **Lemma 13.** *Let $B : \mathbf{Sets} \to \mathbf{Sets}$ be an endofunctor with a final coalgebra (Z, ζ), and let $\mathbf{C}$ be a class of B-coalgebras. If $\mathbf{C}$ is closed under homomorphic images*[5] *and has a final object (E, ε), then $!_\varepsilon : E \to Z$ is injective.*

A *subcoalgebra* of a B-coalgebra (X, β) is an injective map $\iota : U \hookrightarrow X$ such that $\beta|_U$ factors through $B(\iota)$. A B-coalgebra is *locally finite* if every of its states is contained in (the image of) a finite subcoalgebra. We instantiate Lemma 13 in the case where $B = B_M$, $(E, \varepsilon) = (\mathtt{Exp}/{\equiv}, \bar{\epsilon})$, and $\mathbf{C}$ is the class of locally finite B_M-coalgebras. Completeness of $\equiv$ with respect to behavioural equivalence follows shortly after, for if $[\![e]\!] = [\![f]\!]$, then $!_{\bar{\epsilon}}([e]_\equiv) = !_{\bar{\epsilon}}([f]_\equiv)$. By Lemma 13, $!_{\bar{\epsilon}}$ is injective, so $[e]_\equiv = [f]_\equiv$ or equivalently $e \equiv f$. To establish the converse of Theorem 12, it suffices to show that our choices of (E, ε) and $\mathbf{C}$ satisfy the hypotheses of Lemma 13.

Before we continue, we would like to remind the reader of Assumption 3, that S only has operations of finite arity, as up until now it has not been strictly necessary.

▶ **Lemma 14.** *The coalgebra $(\mathtt{Exp}, \epsilon)$ is locally finite.*

Proof. Given $e \in \mathtt{Exp}$, we construct a subcoalgebra of $(\mathtt{Exp}, \epsilon)$ that has a finite set of states that includes e. To this end, define $U : \mathtt{Exp} \to \mathcal{P}_\omega(\mathtt{Exp})$ by

$$U(v) = \{v\} \qquad U(ae) = \{ae\} \cup U(e) \qquad U(\sigma(e_1, \ldots, e_n)) = \{\sigma(e_1, \ldots, e_n)\} \cup \bigcup_{i<n} U(e_i)$$

$$U(\mu v\ e) = \{\mu v\ e\} \cup U(e)[\mu v\ e//v] = \{\mu v\ e\} \cup \{f[\mu v\ e//v] \mid f \in U(e)\}$$

[5] Ie., if $(X, \beta) \in \mathbf{C}$ and $h : (X, \beta) \to (Y, \vartheta)$, then $(h[X], \vartheta|_{h[X]}) \in \mathbf{C}$.

Note that $e \in U(e)$ for all $e \in \mathtt{Exp}$ and that $U(e)$ is finite. We begin with following claim, which says that the outgoing transitions of e are given in terms of expressions from $U(e)$: for any $e \in \mathtt{Exp}$, there is a representative S-term $p \in \epsilon(e)$ such that $p \in S^*(V + A \times U(e))$. This can be seen by induction on the construction of e, the only interesting case being in the inductive step $\mu v\ e$. Here, let $p \in \epsilon(e)$ and observe that $p[\mu v\ e//v]$ is a representative of $\epsilon(\mu v\ e)$ in $S^*(V + A \times U(\mu v\ e))$.

To finish the proof of the lemma, fix an $e \in \mathtt{Exp}$ and define a sequence of sets beginning with $U_0 = \{e\}$ and proceeding with

$$U_{n+1} = U_n \cup \bigcup_{e_0 \in U_n} \{g \mid (\exists a \in A)(\exists p \in \epsilon(e_0) \cap S^*(V + A \times U(e_0)))\ (a, g) \text{ appears in } p\}$$

Then $U_0 \subseteq U_1 \subseteq \cdots \subseteq U(e)$ and the latter set is finite, so $U := \bigcup U_n$ is finite and contained in $U(e)$. We define a coalgebra structure $\epsilon_U : U \to B_M U$ by taking $\epsilon_U(e) = [p]_{\mathsf{E}}$ where if $e \in U_n$, then p is a representative of $\epsilon(e)$ in $S^*(V + A \times U_{n+1})$. Since $S^*(V + A \times U_{n+1}) \subseteq S^*(V + A \times U)$, this defines a B_M-coalgebra structure on U. Where $\iota : U \hookrightarrow \mathtt{Exp}$, we have $\epsilon(\iota(e)) = \epsilon(e) = B_M(\iota)\ \epsilon_U(e)$, so (U, ϵ_U) is a finite subcoalgebra of $(\mathtt{Exp}, \epsilon)$ containing e. ◀

The class of locally finite coalgebras is closed under homomorphic images: if (X, β) is locally finite and $h : (X, \beta) \to (Y, \vartheta)$ is a surjective homomorphism, then for any $y \in Y$ and $x \in X$ such that $h(x) = y$, and for any finite subcoalgebra U of (X, β) containing x, $h[U]$ is a finite subcoalgebra of (Y, ϑ) containing y [21]. Since y was arbitrary, it follows from Lemma 11 that $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$ is locally finite.

What remains to be seen among the hypotheses of Lemma 13 is that $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$ is the *final* locally finite coalgebra, meaning that for any locally finite coalgebra (X, β) there is a unique coalgebra homomorphism $(X, \beta) \to (\mathtt{Exp}/{\equiv}, \bar{\epsilon})$. Every homomorphism from a locally finite coalgebra is the union of its restrictions to finite subcoalgebras, so it suffices to see that every finite subcoalgebra of (X, β) admits a unique coalgebra homomorphism into $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$.

To this end, we make use of an old idea, possibly originating in the work of Salomaa [39]. We associate with every finite coalgebra a certain system of equations whose solutions (in $\mathtt{Exp}/{\equiv}$) are in one-to-one correspondence with coalgebra homomorphisms into $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$. Essentially, if a system admits a unique solution, then its corresponding coalgebra admits a unique homomorphism into $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$. This would then establish finality.

▶ **Definition 15.** *A* (finite) system of equations *is a sequence of the form* $\{x_i = e_i\}_{i \le n}$ *where* $x_i \in V$ *and* $e_i \in \mathtt{Exp}$ *for* $i \le n$*, and none of* $x_1, \ldots, x_n$ *appear as bound variables in any of* $e_1, \ldots, e_n$*. A system of equations* $\{x_i = e_i\}_{i \le n}$ *is* guarded *if* $x_1, \ldots, x_n$ *are guarded in* e_i *for each* $i \le n$*. A* solution *to* $\{x_i = e_i\}_{i \le n}$ *is a function* $\phi : \{x_1, \ldots, x_n\} \to \mathtt{Exp}$ *such that*

$$\phi(x_i) \equiv e_i[\phi(x_1)/x_1, \ldots, \phi(x_n)/x_n]$$

for all $i \le n$ *and* $x_1, \ldots, x_n$ *do not appear free in* $\phi(x_i)$ *for any* $i \le n$*.*

Every finite B_M-coalgebra (X, β) gives rise to a guarded system of equations in the following way: for each $p \in S^*(V + A \times X)$, define $p^\dagger$ inductively as

$$v^\dagger = v \qquad (a, e)^\dagger = ae \qquad \sigma(f_1, \ldots, f_n)^\dagger = \sigma(f_1^\dagger, \ldots, f_n^\dagger)$$

and for each $x \in X$, let p_x be a representative of $\beta(x)$. The[6] system of equations *associated with* (X, β) is then defined to be $\{x = p_x^\dagger\}_{x \in X}$. We treat the elements of X as variables in these equations, and note that by definition every $y \in X$ is guarded in $p_x^\dagger$.

[6] Technically speaking, there could be many systems of equations associated with a given coalgebra. We say "the" system of equations because any two have the same set of solutions up to E.

▶ **Theorem 16.** *Let (X, β) be a finite B_M-coalgebra and $\phi : X \to \mathtt{Exp}$ a function. Then the composition $[-]_{\equiv} \circ \phi : X \to \mathtt{Exp}/{\equiv}$ is a B_M-coalgebra homomorphism if and only if ϕ is a solution to the system of equations associated with (X, β).*

Proof. We begin by observing that $\bar{\epsilon} : \mathtt{Exp}/{\equiv} \to B_M\mathtt{Exp}/{\equiv}$ is a bijection. Indeed, the map $(-)^{\heartsuit} : B_M\mathtt{Exp} \to \mathtt{Exp}/{\equiv}$ defined to be the unique map satisfying

$$[v]_{\mathsf{E}}^{\heartsuit} = [v]_{\equiv} \qquad [(a, e)]_{\mathsf{E}}^{\heartsuit} = [ae]_{\equiv} \qquad [\sigma(p_1, \ldots, p_n)]_{\mathsf{E}}^{\heartsuit} = \sigma([p_1]_{\mathsf{E}}^{\heartsuit}, \ldots, [p_n]_{\mathsf{E}}^{\heartsuit})$$

is its inverse. By construction, $\bar{\epsilon}([p]_{\mathsf{E}}^{\heartsuit}) = [p]_{\mathsf{E}}$ for any $p \in S^*(V + A \times \mathtt{Exp})$, so it suffices to see that $\bar{\epsilon}([e]_{\equiv})^{\heartsuit} = [e]_{\equiv}$ for all $e \in \mathtt{Exp}$. This can be done by induction on the construction of e, and again the only interesting case is $\mu v\ e$. For this case, observe that

$$\begin{aligned}
\bar{\epsilon}([\mu v\ e]_{\equiv})^{\heartsuit} &= (B_M([-]_{\equiv})(\epsilon(e)[\mu v\ e//v]))^{\heartsuit} \\
&= (B_M([-]_{\equiv})(\epsilon(e[\mu v\ e//v])))^{\heartsuit} \\
&= (\bar{\epsilon}([e[\mu v\ e//v]]_{\equiv}))^{\heartsuit} && ([-]_{\equiv} \text{ a coalg. homom.}) \\
&= (\bar{\epsilon}([e]_{\equiv})[[\mu v\ e]_{\equiv}//v])^{\heartsuit} \\
&= \bar{\epsilon}([e]_{\equiv})^{\heartsuit}[[\mu v\ e]_{\equiv}//v] && (*) \\
&= [e[\mu v\ e//v]]_{\equiv} && (\text{induct. hyp.}) \\
&= [\mu v\ e]_{\equiv} && (\mathsf{R1})
\end{aligned}$$

We have used several properties of syntactic substitution in the above calculation; see [42, Appendix C] for details. We have also used that guarded syntactic substitution commutes with $(-)^{\heartsuit}$ in (*), but this follows from an easy induction on terms in $S^*(V + A \times (\mathtt{Exp}/{\equiv}))$.

Now let $\{x = p_x^{\dagger}\}_{x \in X}$ be the system of equations associated with the coalgebra (X, β). Observe that for any $x, y \in X$, if y appears in p_x, then it is guarded in $p_x^{\dagger}$. This means that $\phi : X \to \mathtt{Exp}$ is a solution to $\{x = p_x^{\dagger}\}_{x \in X}$ if and only if $\phi(x) \equiv p_x^{\dagger}[\phi(y)//y]_{y \in X}$. Now, if $\beta(x) = [p_x]_{\mathsf{E}}$, we see that

$$\begin{aligned}
(B_M([-]_{\equiv} \circ \phi)(\beta(x)))^{\heartsuit} &= (B_M([-]_{\equiv}) \circ B_M(\phi)([p_x]_{\mathsf{E}}))^{\heartsuit} && (\text{functoriality}) \\
&= (B_M([-]_{\equiv})([p_x]_{\mathsf{E}}[\phi(y)//y]_{y \in X}))^{\heartsuit} && (B_M(\phi) \text{ an alg. homom.}) \\
&= ([p_x]_{\mathsf{E}}[[\phi(y)]_{\equiv}//y]_{y \in X})^{\heartsuit} \\
&= [p_x^{\dagger}[\phi(y)//y]_{y \in X}]_{\equiv}
\end{aligned}$$

Thus, ϕ is a solution to the system $\{x = p_x^{\dagger}\}_{x \in X}$ if and only if

$$[-]_{\equiv} \circ \phi(x) = (-)^{\heartsuit} \circ B_M([-]_{\equiv} \circ \phi) \circ \beta(x) \tag{3}$$

for every $x \in X$. The maps $(-)^{\heartsuit}$ and $\bar{\epsilon}$ are inverse to one another, so Equation (3) is equivalent to the identity $\bar{\epsilon} \circ [-]_{\equiv} \circ \phi = B_M([-]_{\equiv} \circ \phi) \circ \beta$. This identity is the defining property of a coalgebra homomorphism of the form $[-]_{\equiv} \circ \phi$. ◀

As a direct consequence of Theorem 16, we see that a finite subcoalgebra $U \hookrightarrow \mathtt{Exp}$ of $(\mathtt{Exp}, \epsilon)$ is a solution to the system of equations associated with $(U, \epsilon|_U)$.

▶ **Example 17.** The system of equations associated with the automaton in Example 7 is the two-element set $\{x_1 = a_1x_2 +_b u, x_2 = v +_b a_2x_1\}$. The map $\phi : \{x_1, x_2\} \to \mathtt{Exp}$ defined by $\phi(x_1) = \mu w\ (a_1(v +_b a_2w) +_b u)$ and $\phi(x_2) = v +_b a_2\ \phi(x_1)$ is a solution.

Theorem 16 establishes a one-to-one correspondence between solutions to systems and coalgebra homomorphisms as follows. Say that two solutions ϕ and ψ to a system $\{x_i = e_i\}_{i \leq n}$ are $\equiv$-*equivalent* if $\phi(x_i) \equiv \psi(x_i)$ for all $i \leq n$. Starting with a solution $\phi : X \to \mathtt{Exp}$ to the system associated with (X, β), we obtain the homomorphism $[-]_\equiv \circ \phi$ using Theorem 16. A pair of solutions ϕ and ψ are $\equiv$-equivalent if and only if $[-]_\equiv \circ \phi = [-]_\equiv \circ \psi$, so up to $\equiv$-equivalence the correspondence $\phi \mapsto [-]_\equiv \circ \phi$ is injective. Going in the opposite direction and starting with a homomorphism $\psi : (X, \beta) \to (\mathtt{Exp}/{\equiv}, \bar{\epsilon})$, let e_x be a representative of $\psi(x)$ for each $x \in X$ and define $\phi := \lambda x.e_x$. Then ϕ is a solution to (X, β), and $[-]_\equiv \circ \phi = \psi$. Thus, up to $\equiv$-equivalence, solutions to systems are in one-to-one correspondence with coalgebra homomorphisms into $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$.

Say that a system *admits a unique solution up to* $\equiv$ if it has a solution and any two solutions to the system are $\equiv$-equivalent. Since, up to $\equiv$-equivalence, solutions to a system associated with a coalgebra (X, β) are in one-to-one correspondence with coalgebra homomorphisms $(X, \beta) \to (\mathtt{Exp}/{\equiv}, \bar{\epsilon})$, it suffices for the purposes of satisfying the hypotheses of Lemma 13 to show that every finite guarded system of equations admits a unique solution up to $\equiv$. The following theorem is a generalisation of [30, Theorem 5.7].

▶ **Theorem 18.** *Every finite guarded system of equations admits a unique solution up to* $\equiv$.

The proof is a recreation of the one that appears under [30, Theorem 5.7] with the more general context of our paper in mind. Remarkably, the essential details of the proof remain unchanged despite the jump in the level of abstraction between the two results.

Completeness of $\equiv$ with respect to behavioural equivalence is now a direct consequence of Lemma 13 and Theorems 16 and 18.

▶ **Corollary 19** (Completeness). *Let* $e, f \in \mathtt{Exp}$. *If* $[\![e]\!] = [\![f]\!]$, *then* $e \equiv f$.

One way to interpret this theorem is that the algebra $(\mathtt{Exp}/{\equiv}, \hat{\alpha})$ of process terms modulo $\equiv$ is isomorphic to a subalgebra of (Z, γ), or dually $(\mathtt{Exp}/{\equiv}, \bar{\epsilon})$ is a subcoalgebra of (Z, ζ). It is in this sense that ARB, ACF, APA, and ANP are algebras of behaviours.

6 Star Fragments

In this section we study a fragment of our specification languages consisting of *star expressions*. These include primitive actions from A, a form of sequential composition, and analogues of the Kleene star. We do not aim to give a complete axiomatisation of behavioural equivalence for star expressions, as even in simple cases this is notoriously difficult. Nevertheless, we think it is valuable to extrapolate from known examples a speculative axiomatisation independent of the specification languages from previous sections.

Fix an algebraic theory (S, E) and assume S consists of only constants and binary operations. Its *star fragment* is the set $\mathtt{SExp}$ of expressions given by the grammar

$$e, e_i ::= c \mid \mathsf{1} \mid a \mid e_1 +_\sigma e_2 \mid e_1 e_2 \mid e^{(\sigma)}$$

where $a \in A$, c is a constant in S, and σ is a binary S-operation.

The star fragment of an algebraic theory is a fragment of $\mathtt{Exp}$ in the sense that star expressions can be thought of as shorthands for process terms, as we explain next. In this translation, we fix a distinguished variable $\underline{u} \in V$, called the *unit*, which will denote successful termination, and we also fix a variable v distinct from the unit, which will appear in the fixpoint. The translation of star expressions to process terms is defined to be

$$\mathsf{1} \mapsto \underline{u} \qquad a \mapsto a\underline{u} \qquad e_1 +_\sigma e_2 \mapsto \sigma(e_1, e_2) \qquad e_1 e_2 \mapsto e_1[e_2/\underline{u}] \qquad e^{(\sigma)} \mapsto \mu v\, (e[v/\underline{u}] +_\sigma \underline{u})$$

$$\ell(c) = [c]_{\mathsf{E}} \qquad \ell(e_1 +_\sigma e_2) = \sigma(\ell(e_1), \ell(e_2))$$
$$\ell(1) = [\checkmark]_{\mathsf{E}} \qquad \ell(ef) = p(\ell(f), [(a_1, e_1 f)]_{\mathsf{E}}, \dots, [(a_n, e_n f)]_{\mathsf{E}})$$
$$\ell(a) = [(a, 1)]_{\mathsf{E}} \qquad \ell(e^{(\sigma)}) = p([0]_{\mathsf{E}}, [(a_1, e_1 e^{(\sigma)})]_{\mathsf{E}}, \dots, [(a_n, e_n e^{(\sigma)})]_{\mathsf{E}}) +_\sigma [\checkmark]_{\mathsf{E}}$$

Figure 3 The coalgebra structure map $\ell : \mathtt{SExp} \to L_M \mathtt{SExp}$. Here, c is a constant of S, σ is a binary operation of S, $a \in A$, and $e, e_i \in \mathtt{SExp}$. In the last two equations, $\ell(e) = [p(\checkmark, (a_1, e_1), \dots, (a_n, e_n))]_{\mathsf{E}}$ for some $p \in S^*(\{\checkmark\} + A \times \mathtt{SExp})$.

Sequential composition of terms is associative and distributes over branching operations on the right-hand side[7]: for any $e_1, e_2, f \in \mathtt{SExp}$, $(e_1 +_\sigma e_2)f$ and $e_1 f +_\sigma e_2 f$ translate to the same process term. Similarly, the intuitively correct identities $\mathtt{1}e = e = e\mathtt{1}$ hold modulo translation, as well as the identity $0e = 0$.[8]

The operational semantics for star expressions is given by an L_M-coalgebra $(\mathtt{SExp}, \ell)$ in Figure 3, where $L_M = M(\{\checkmark\} + A \times \mathrm{Id})$. Abstractly, the operational interpretation $\ell(e)$ of a star expression e is obtained by translating e into a process term (also called e) and then identifying $\underline{a}$ with $\checkmark$ in $\epsilon(e)$. While the notation is somewhat opaque at this level of generality, in specific instances the map ℓ amounts to a familiar transition structure.

▶ **Example 20.** The star fragment of ACF from Example 3 and Example 7 coincides with GKAT, the algebra of programs introduced in [24] and studied further in [44, 40]. Instantiating $\mathtt{SExp}$ in this context reveals the syntax

$$e_i ::= 0 \mid \mathtt{1} \mid a \mid e_1 +_b e_2 \mid e_1 e_2 \mid e^{(b)}$$

for $b \subseteq \mathtt{At}$ and $a \in A$. This is nearly the syntax of GKAT, the only difference being the presence of $\mathtt{1}$ and 0 instead of Boolean constants $b \subseteq \mathtt{At}$. This is merely cosmetic, as we can just as well define $b := \mathtt{1} +_b 0$.

In this context, $M = (1 + \mathrm{Id})^{\mathtt{At}}$, and so $L_M \cong (2 + A \times \mathrm{Id})^{\mathtt{At}}$, which is the precise coalgebraic signature of the automaton models of GKAT expressions. It is readily checked that the operational semantics of GKAT also coincides with the operational semantics of the star fragment of ACF given above.

▶ **Example 21.** The star fragment of APA from Example 4 and Example 8 is a subset of the calculus of programs introduced in [11], but with an iteration operator for each $p \in [0, 1]$. Instantiating $\mathtt{SExp}$ in this context reveals the syntax

$$e_i ::= 0 \mid \mathtt{1} \mid a \mid e_1 +_p e_2 \mid e_1 e_2 \mid e^{(p)}$$

for $p \in [0, 1]$ and $a \in A$. The process $e^{(p)}$ can be thought of as a generalised Bernoulli process that runs e until it reaches $\checkmark$ and then flips a weighted coin to decide whether to start from the beginning of e or to terminate successfully.

We now provide a candidate axiomatisation for the star fragment while leaving the question of completeness open. Say that a star expression e is *guarded* if the unit is guarded in e as an expression in $\mathtt{Exp}$. We define E^* to be the theory consisting of E, the axiom schema

$$(\mathsf{E}^*1) \quad 1e = e1 = e \qquad (\mathsf{E}^*3) \quad e_1(e_2 e_3) = (e_1 e_2)e_3$$
$$(\mathsf{E}^*2) \quad ce = c \qquad (\mathsf{E}^*4) \quad (e +_\tau 1)^{(\sigma)} = (e +_\tau 0)^{(\sigma)}$$

[7] But not on the left-hand side! Observe the difference between the processes $a(b + c)$ and $ab + ac$ here.
[8] But not $e0 = 0$! See also the previous footnote.

and the inference rules

$$(\mathsf{E}^*5)\quad \frac{e \text{ is guarded}}{e^{(\sigma)} = ee^{(\sigma)} +_\sigma 1} \qquad (\mathsf{E}^*6)\quad \frac{g = eg +_\sigma f \qquad e \text{ is guarded}}{g = e^{(\sigma)} f}$$

In the specific cases where $\mathsf{E} = \mathsf{SL}$ and $\mathsf{E} = \mathsf{GS}$, E^* is equivalent to the candidate axiomatisations for the star fragments of ARB [30] and ACF [44, 40].[9]

There is a difference between our axioms and the axioms in [30, 44, 40]: instead of (E^*5), all equations of the form $e^{(\sigma)} = ee^{(\sigma)} +_\sigma 1$ appear in loc cit, even those where e is unguarded. We adopt (E^*5) instead because the unrestricted version of (E^*5) fails to be sound for the star expressions of APA. For example, if $e = 1 +_{\frac{1}{3}} a$, then $ee^{(1/2)} +_{\frac{1}{2}} 1 \xrightarrow{7/12} \checkmark$ while $e^{(\frac{1}{2})} \xrightarrow{1/2} \checkmark$.

We are confident that a completeness result can be obtained in several instances of the framework for the axiomatisation we have suggested above. However, in several cases this cannot happen. For example, there is no way to derive the identity $((a +_{\frac{1}{2}} 1) + b)^* = ((a +_{\frac{1}{2}} 0) + b)^*$ from CS^* (see Example 5) despite these expressions being behaviourally equivalent. What is likely missing from CS is a number of axioms that would allow 1 to be moved to the top level of every S-term (and then replaced by 0 using (E^*4)). Algebraic theories where this is doable are called *skew-associative*, which we define formally as follows.

▶ **Definition 22.** *An algebraic theory (S, E) consisting of constants and binary operations is called* skew-associative *if for any pair of binary operations σ_1, τ_1, there is a pair of binary operations σ_2, τ_2 such that $\sigma_1(x, \tau_1(y, z)) = \tau_2(\sigma_2(x, y), z)$ appears in E.*

Many of the examples we care about are skew-associative, including the theories of semilattices, guarded semilattices, and convex algebras.

▶ **Question 1.** *Assume (S, E) is a skew-associative algebraic theory. If e and f are behaviourally equivalent star expressions, is it true that $\mathsf{E}^* \vdash e = f$?*

7 Related Work

Our framework can be seen as a generalisation of Milner's ARB [30] that reaches beyond nondeterministic choice and covers several other process algebras already identified in the literature. For example, instantiating our framework in the theory of pointed convex algebras produces the algebra we have called APA (see Example 8), which only differs from the algebra PE of Stark and Smolka [45] in the axiom (R1). In loc cit, the requirement that the variable be guarded in the recursed expression is absent because recursion is computed as a least fixed point in their semantics. This is not how we interpret recursion. We have included the guardedness requirement because it is necessary for the soundness of the axiom in our semantics: for example, where $e = u +_{\frac{1}{2}} v$, we have $\mu v\ e \xrightarrow{1/2} u$ and $e[\mu v\ e/v] \xrightarrow{3/4} u$. In contrast, both $\mu v\ e$ and $e[\mu v\ e/v]$ exit in u with probability 1 in [45].

For another example, instantiating our framework in the theory CS of pointed convex semilattices gives ANP (see Example 5), which differs from the calculus of Mislove, Ouaknine, and Worrell [31] on two points. Firstly, their axiomatisation contains an unguarded version of (R1), like in [45]. Secondly, the underlying algebraic theory of [31] corresponds to CS extended with the axiom $x +_p 0 = 0$. The resulting theory is known in the literature as that of *convex semilattices with top* [11].

[9] See [42, Appendix F] for details.

Star expressions for non-deterministic processes appeared in the work of Milner [30] as a fragment of ARB and can be thought of as a bisimulation-focused analogue of Kleene's regular expressions for NFAs. While the syntaxes of Milner's star expressions and Kleene's regular expressions are the same, there are several important differences between their interpretations. For example, sequential composition is interpreted as the variable substitution $ef := e[f/\underline{u}]$ in Milner's paper, which fails to distribute over $+$ on the left. A notable insight from [30] is that, despite these differences, an iteration operator $(-)^*$ can be defined for Milner's star expressions that satisfies many of the same identities as the Kleene star. Given a variable v distinct from the unit and a process term e of ARB in which at most the unit is free,

$$e^* = \mu v\ (e[v/\underline{u}] + \underline{u})$$

defines the iteration operator in Milner's star fragment of ARB. In Section 6, we generalised this construction of Milner for the more general process types that we considered in this paper. Our proposed axiomatization is also inspired by Milner's work. We expect completeness of our general calculus will be a hard problem, as completeness in the instantiation to ARB was open for decades despite the extensive literature on the subject [15, 14, 16, 3, 20].

There are clear parallels between our work and the thesis of Silva [43], in which a family of calculi is introduced that includes one-exit versions of ARB, ACF, and APA (see Examples 2, 7, and 8). The main difference is that our framework is parametric on a finitary monad on **Sets** whereas Silva's is centered around one particular theory (semilattices). However, her work considers general polynomial functors on **Sets**, which we have not yet done in our paper. We could achieve a similar level of generality by replacing $A \times \mathrm{Id}$ in our signatures Σ_M, B_M, and L_M with an arbitrary polynomial functor.

Our results are also in the same vein as the work of Myers on coalgebraic expressions [32]. Coalgebraic expressions generalise the calculi of [43] to arbitrary finitary coalgebraic signatures on a variety of algebras, and furthermore have totally defined recursion operators similar to ours. However, the focus of the framework of coalgebraic expressions is on language semantics, achieved by lifting the coalgebraic signature to a variety. This distinguishes the framework from our approach: we focus on bisimulation semantics. This focus is also the reason we interpret our B_M-coalgebras in **Sets** and not in the Kleisli category of the monad M, as is done in [22] to capture trace semantics of coalgebras.

Finally, there is also a notable connection to the iterative theories of Elgot [13, 6, 7, 33]. Theorem 18 in particular implies that our process algebras are examples of iterative algebras.

8 Future Work

In this paper, we introduced a family of process types whose branching structure is determined by an algebraic theory. We provided each process type with a fully expressive specification language paired with a sound and complete axiomatisation of behavioural equivalence.

There are several instantiations of our framework that we have not yet explored and are of interest. For example, processes with multiset branching given by the theory of commutative monoids produces nondeterministic processes with a simplistic notion of resources. Another example is nondeterministic weighted processes with branching captured by the monad arising from the weak distributive law between the free semimodule and powerset monads [9]. Yet another instantiation arises from the theory of monoids (presenting the list monad), which produces processes related to breadth-first search algorithms.

Star fragments offer a uniform construction of Kleene-like algebras for a variety of paradigms of computing. However, our framework does not suggest an axiomatisation of the star fragment that combines nondeterministic and probabilistic choice, as the theory CS is

not skew-associative (see Definition 22). We would like to expand our framework to include this fragment as it provides an interesting but nonstandard interpretation of a part of the language ProbNetKAT used to verify probabilistic networks [17].

We would also like to investigate the question at the end of Section 6 of whether E^* is complete for skew-associative theories. In particular, we believe that a connection can be made to the work of Grabmayer and Fokkink [20] on LLEE-charts, which provides a completeness theorem for the so-called 1-free expressions of the star fragment of ARB. Our process algebras also have uniformly defined 1-free star fragments, and it is not difficult to give 1-free versions of the axiomatisation E^*. We intend to suitably generalise LLEE-charts to arbitrary skew-associative theories and prove completeness theorems for 1-free star fragments.

Finally, we would like to know whether our operational semantics for process terms is an instance of the mathematical operational semantics introduced by Turi and Plotkin [48].

References

1 Suzana Andova. Process algebra with probabilistic choice. In Joost-Pieter Katoen, editor, *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop, ARTS'99, Bamberg, Germany, May 26-28, 1999. Proceedings*, volume 1601 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 1999. doi:10.1007/3-540-48778-6_7.

2 Jos C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005. doi:10.1016/j.tcs.2004.07.036.

3 Jos C. M. Baeten, Flavio Corradini, and Clemens Grabmayer. On the star height of regular expressions under bisimulation (extended abstract), 2006.

4 Michael Barr. Terminal coalgebras in well-founded set theory. *Theor. Comput. Sci.*, 114(2):299–315, 1993. doi:10.1016/0304-3975(93)90076-6.

5 Jan A. Bergstra and Jan Willem Klop. Process theory based on bisimulation semantics. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop, Noordwijkerhout, The Netherlands, May 30 - June 3, 1988, Proceedings*, volume 354 of *Lecture Notes in Computer Science*, pages 50–122. Springer, 1988. doi:10.1007/BFb0013021.

6 Stephen L. Bloom and Calvin C. Elgot. The existence and construction of free iterative theories. *J. Comput. Syst. Sci.*, 12(3):305–318, 1976. doi:10.1016/S0022-0000(76)80003-7.

7 Stephen L. Bloom and Zoltán Ésik. Varieties of iteration theories. *SIAM J. Comput.*, 17(5):939–966, 1988. doi:10.1137/0217059.

8 Stephen L. Bloom and Ralph Tindell. Varieties of "if-then-else". *SIAM J. Comput.*, 12(4):677–707, 1983. doi:10.1137/0212047.

9 Filippo Bonchi and Alessio Santamaria. Combining semilattices and semimodules. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12650 of *Lecture Notes in Computer Science*, pages 102–123. Springer, 2021. doi:10.1007/978-3-030-71995-1_6.

10 Filippo Bonchi, Alexandra Silva, and Ana Sokolova. The power of convex algebras. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPIcs*, pages 23:1–23:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CONCUR.2017.23.

11 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The theory of traces for systems with nondeterminism and probability. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–14. IEEE, 2019. doi:10.1109/LICS.2019.8785673.

12 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. Presenting Convex Sets of Probability Distributions by Convex Semilattices and Unique Bases. In Fabio Gadducci and Alexandra Silva, editors, *9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021)*, volume 211 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CALCO.2021.11.

13 Calvin C. Elgot. Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*, pages 175–230. Elsevier, 1975. doi:10.1016/S0049-237X(08)71949-9.

14 Wan Fokkink. Axiomatizations for the perpetual loop in process algebra. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming*, pages 571–581, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

15 Wan J. Fokkink and Hans Zantema. Basic process algebra with iteration: Completeness of its equational axioms. *Comput. J.*, 37(4):259–268, 1994. doi:10.1093/comjnl/37.4.259.

16 Wan J. Fokkink and Hans Zantema. Termination modulo equations by abstract commutation with an application to iteration. *Theor. Comput. Sci.*, 177(2):407–423, 1997. doi:10.1016/S0304-3975(96)00254-X.

17 Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic netkat. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 282–309. Springer, 2016. doi:10.1007/978-3-662-49498-1_12.

18 Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *J. ACM*, 24(1):68–95, 1977. doi:10.1145/321992.321997.

19 Clemens Grabmayer. A coinductive version of milner's proof system for regular expressions modulo bisimilarity. In Fabio Gadducci and Alexandra Silva, editors, *9th Conference on Algebra and Coalgebra in Computer Science, CALCO 2021, August 31 to September 3, 2021, Salzburg, Austria*, volume 211 of *LIPIcs*, pages 16:1–16:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CALCO.2021.16.

20 Clemens Grabmayer and Wan J. Fokkink. A complete proof system for 1-free regular expressions modulo bisimilarity. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 465–478. ACM, 2020. doi:10.1145/3373718.3394744.

21 H. Peter Gumm. Elements of the general theory of coalgebras. *LUATCS'99, Rand Afrikaans University, Johannesburg*, 1999.

22 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace semantics via coinduction. *Log. Methods Comput. Sci.*, 3(4), 2007. doi:10.2168/LMCS-3(4:11)2007.

23 Bart Jacobs. Convexity, duality and effects. In Cristian S. Calude and Vladimiro Sassone, editors, *Theoretical Computer Science - 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, volume 323 of *IFIP Advances in Information and Communication Technology*, pages 1–19. Springer, 2010. doi:10.1007/978-3-642-15240-5_1.

24 Dexter Kozen and Wei-Lung Dustin Tseng. The böhm-jacopini theorem is false, propositionally. In Philippe Audebaud and Christine Paulin-Mohring, editors, *Mathematics of Program Construction, 9th International Conference, MPC 2008, Marseille, France, July 15-18, 2008. Proceedings*, volume 5133 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2008. doi:10.1007/978-3-540-70594-9_11.

25 Joachim Lambek. A fixpoint theorem for complete categories. *Mathematische Zeitschrift*, 103(2):151–161, 1968.

26 Ernest G. Manes. *Algebraic Theories*. Graduate Texts in Mathematics. Springer-Verlag New York, 1 edition, 1976. doi:10.1007/978-1-4612-9860-1.

27 Ernest G. Manes. Equations for if-then-else. In Stephen D. Brookes, Michael G. Main, Austin Melton, Michael W. Mislove, and David A. Schmidt, editors, *Mathematical Foundations of Programming Semantics, 7th International Conference, Pittsburgh, PA, USA, March 25-28, 1991, Proceedings*, volume 598 of *Lecture Notes in Computer Science*, pages 446–456. Springer, 1991. doi:10.1007/3-540-55511-0_23.

28 John McCarthy. A basis for a mathematical theory of computation, preliminary report. In Walter F. Bauer, editor, *Papers presented at the 1961 western joint IRE-AIEE-ACM computer conference, IRE-AIEE-ACM 1961 (Western), Los Angeles, California, USA, May 9-11, 1961*, pages 225–238. ACM, 1961. doi:10.1145/1460690.1460715.

29 Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980. doi:10.1007/3-540-10235-3.

30 Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984. doi:10.1016/0022-0000(84)90023-0.

31 Michael W. Mislove, Joël Ouaknine, and James Worrell. Axioms for probability and nondeterminism. In Flavio Corradini and Uwe Nestmann, editors, *Proceedings of the 10th International Workshop on Expressiveness in Concurrency, EXPRESS 2003, Marseille, France, September 2, 2003*, volume 96 of *Electronic Notes in Theoretical Computer Science*, pages 7–28. Elsevier, 2003. doi:10.1016/j.entcs.2004.04.019.

32 Robert S. R. Myers. Coalgebraic expressions. In Ralph Matthes and Tarmo Uustalu, editors, *6th Workshop on Fixed Points in Computer Science, FICS 2009, Coimbra, Portugal, September 12-13, 2009*, pages 61–69. Institute of Cybernetics, 2009. URL: http://cs.ioc.ee/fics09/proceedings/contrib8.pdf.

33 Evelyn Nelson. Iterative algebras. *Theor. Comput. Sci.*, 25:67–94, 1983. doi:10.1016/0304-3975(83)90014-2.

34 Gordon D. Plotkin. A structural approach to operational semantics. *J. Log. Algebraic Methods Program.*, 60-61:17–139, 2004.

35 D. Pumplün and Helmut Röhrl. Convexity theories IV. klein-hilbert parts in convex modules. *Appl. Categorical Struct.*, 3(2):173–200, 1995. doi:10.1007/BF00877635.

36 Jan J. M. M. Rutten. Automata and coinduction (an exercise in coalgebra). In Davide Sangiorgi and Robert de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*, pages 194–218. Springer, 1998. doi:10.1007/BFb0055624.

37 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

38 Jan J. M. M. Rutten and Daniele Turi. On the foundations of final semantics: Non-standard sets, metric spaces, partial orders. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Semantics: Foundations and Applications*, pages 477–530, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

39 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966. doi:10.1145/321312.321326.

40 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded kleene algebra with tests: Coequations, coinduction, and completeness. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*, volume 198 of *LIPIcs*, pages 142:1–142:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.142.

41 Todd Schmid, Jurriaan Rot, and Alexandra Silva. On star expressions and coalgebraic completeness theorems. In Ana Sokolova, editor, *Proceedings 37th Conference on Mathematical Foundations of Programming Semantics, MFPS 2021, Hybrid: Salzburg, Austria and Online, 30th August - 2nd September, 2021*, volume 351 of *EPTCS*, pages 242–259, 2021. doi:10.4204/EPTCS.351.15.

42 Todd Schmid, Wojciech Rozowski, Alexandra Silva, and Jurriaan Rot. Processes parametrised by an algebraic theory (with appendix), 2022. arXiv:2202.06901.

43 Alexandra Silva. *Kleene coalgebra*. PhD thesis, University of Nijmegen, 2010.

44 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded kleene algebra with tests: verification of uninterpreted programs in nearly linear time. *Proc. ACM Program. Lang.*, 4(POPL):61:1–61:28, 2020. doi:10.1145/3371129.

45 Eugene W. Stark and Scott A. Smolka. A complete axiom system for finite-state probabilistic processes. In Gordon D. Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 571–596. The MIT Press, 2000.

46 Marshall Harvey Stone. Postulates for the barycentric calculus. *Annali di Matematica Pura ed Applicata*, 29(1):25–30, 1949.

47 Tadeusz Świrszcz. Monadic functors and convexity. *Bulletin de L'Académie Polonaise des Sciences*, XXII(1):39–42, 1974.

48 Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 280–291. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614955.

49 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Mathematical Structures in Computer Science*, 16(1):87–113, 2006. doi:10.1017/S0960129505005074.