Instances of computational effects: an algebraic perspective

Sam Staton
Computer Laboratory, University of Cambridge, UK

***Abstract*—We investigate the connections between computational effects, algebraic theories, and monads on functor categories. We develop a syntactic framework with variable binding that allows us to describe equations between programs while taking into account the idea that there may be different instances of a particular computational effect. We use our framework to give a general account of several notions of computation that had previously been analyzed in terms of monads on presheaf categories: the analysis of local store by Plotkin and Power; the analysis of restriction by Pitts; and the analysis of the pi calculus by Stark.**

I. Introduction

This paper is about a framework for axiomatizing equations between effectful computer programs. We are particularly concerned with effects that can be understood in terms of many parametric instances of a simpler effect. For example, when reasoning about local store we can think of the different memory cells each as different instances of the same notion of memory cell. When reasoning about the π-calculus we can think of the many communication channels each as different instances of a single communication channel. Over the past 15 years, several authors have investigated these phenomena in an algebraic way in order to define computational monads. This paper provides a formal algebraic framework which allows us to analyze these kinds of effect in a modular way.

A. Algebraic theories, monads and computational effects

Consider the simple effect of reading a binary digit from memory. This can be described algebraically by saying that the set of computations $\mathfrak{M}$ is equipped with a binary operation $? : \mathfrak{M} \times \mathfrak{M} \to \mathfrak{M}$ satisfying two equations:

$$x \mathbin{?} x \equiv x \qquad\qquad (v \mathbin{?} x) \mathbin{?} (y \mathbin{?} z) \equiv v \mathbin{?} z. \qquad (1)$$

The idea is that $x \mathbin{?} y$ is a computation that first reads the bit and then proceeds as x or as y depending on the result. The first equation says that if you ignore the result then the read is not observable. The second equation says that you get the same result no matter how many times you test.

This theory (1) has been studied by universal algebraists under the name 'rectangular band'. But its relevance to computation is made especially clear when we follow Plotkin and Power [49] in recalling that the free models of the algebraic theory form a monad on the category of sets, and hence a notion of computation in the sense of Moggi [42]. Recall that the free model of an algebraic theory over a set X can be built by first forming the set of all formal expressions built from the operation $?$ and elements of X, and then taking equivalence classes modulo the equations (1). In particular the free rectangular band on a set X is isomorphic to the set of all functions $\{0,1\} \to X$: the 'reader monad'. Thus algebra arises as a technique for defining monads. Moreover, algebra provides a way to explore the space of computational effects, by considering different equations and different ways of combining computational effects.

Instances of computational effects: In this paper we extend the algebraic approach to computational effects to the situation where there are many instances of a particular computational effect. For example, there may be many memory cells: each one is an instance of the theory of reading a bit. We may also build a theory of reading a bit from a communication channel by omitting the two equations (1), so that multi-channel communication can be understood in terms of many instances of the theory for one channel. There are often facilities for creating new instances: allocating a new memory cell or defining a new private communication channel.

The problem of providing an axiomatic account of local instances of effects dates back to Moggi's initial writings about monads [41, §4.1.4]. Over the last 15 years authors have given explanations of local store [49], restriction [47] and the π-calculus ([55], [57]) in ways that are ad hoc but informally inspired by algebra. We put these analyses on a formal foundation by exhibiting an algebraic framework with the following properties:

1) our framework is syntactic, with a deduction system;
2) each algebraic theory determines an enriched monad, so we can build models of Moggi's metalanguage;
3) there are mechanisms for reasoning about different instances of theories and introducing new instances;
4) the framework encompasses earlier analyses of local store [49], restriction [47] and the π-calculus [55].

The development of the framework is guided by the categorical foundation of enriched algebraic theories [30], building on recent developments by Lack and Rosický [32], Melliès et al. ([37], [38], [8]), Power [51] and others.

B. Algebraic theories for instances of effects

If we have many instances of the notion of reading a bit from memory, then we have a binary operation $?_a$ parameterized in the instance a. A model of the theory now has an operation $? : \mathfrak{M} \times \alpha \times \mathfrak{M} \to \mathfrak{M}$, where $\mathfrak{M}$ is a set of all computations and α is a set of memory cells. Similar parameterization can be done for reading from communication channels. We call this a *parameterized algebraic theory*.

In the setting of memory cells, it is reasonable to add a law asserting that different instances commute with each other:

$$(u \mathbin{?_a} v) \mathbin{?_b} (x \mathbin{?_a} y) \equiv (u \mathbin{?_b} x) \mathbin{?_a} (v \mathbin{?_b} y)$$

In Standard ML this would be written

```
let val c = !a in let val d = !b in (c,d) end end
≡ let val d = !b in let val c = !a in (c,d) end end
```

In general, commutativity of instances requires some care. Consider the effect of writing to memory: our set of computations has two operations $\mathrm{w}_0, \mathrm{w}_1 : \alpha \times \mathfrak{M} \to \mathfrak{M}$, so that $\mathrm{w}_0(a, x)$ and $\mathrm{w}_1(a, x)$ are interpretations of the ML expressions $a{:=}0\,; x$ and $a{:=}1\,; x$ respectively. Assignments to different locations commute, but assignments to the same location do not: $(a{:=}0\,; b{:=}1\,; x) \equiv (b{:=}1\,; a{:=}0\,; x)$ provided $a \neq b$. We can write this as an unconditional program equation

$$a{:=}0\,; b{:=}1\,; x \;\equiv\; \texttt{if}\ a{=}b\ \texttt{then}\ a{:=}1\,; x\ \texttt{else}\ b{:=}1\,; a{:=}0\,; x$$

provided that our set α of instances permits equality testing.

a) Axiomatization of equality types (§IV): The idea of an equality type is built in to Standard ML and described via type classes in Haskell. Informally it is achieved by a function $= : \alpha \times \alpha \to \mathit{bool}$, or expressed in continuation passing style as $?_= : \mathfrak{M} \times \alpha \times \alpha \times \mathfrak{M} \to \mathfrak{M}$, with the idea that $x \mathbin{?_{a=b}} y$ is the computation that first tests whether a and b are equal; if so, continue as x, but if not, continue as y.

In this paper we study program equations for equality testing. There are four laws, including $x \mathbin{?_{a=a}} y \equiv x$ (equality testing of identical instances always succeeds) and $x(a) \mathbin{?_{a=b}} y \equiv x(b) \mathbin{?_{a=b}} y$ (equality is substitutive). Notice that the second law is higher order. The idea of using an axiomatization of equality testing in this way stems from the work of Stark [57] and others.

b) Restriction and instance generation (§V): We also study the program equations for restricting and generating instances. This includes allocating a new memory cell or a new communication channel. We do this algebraically using a second-order function symbol $\nu : \mathfrak{M}^\alpha \to \mathfrak{M}$, so that $\nu(a.x(a))$ is a computation that first allocates a new instance a and then proceeds as x. In Standard ML it would be written `let val a=ref(0) in x a end`, and in Haskell `newSTRef 0 >>= x`. We can axiomatize the property that 'the allocated cell is fresh' in an equational way by using the equality testing operation: $\nu b.(x(b) \mathbin{?_{a=b}} y(b)) \equiv \nu b.y(b)$.

C. Analysis of models of algebraic theories

We analyze the notions of program equivalence that we have axiomatized in three ways: by giving an account of model theory; by studying representations; and by giving a precise comparison with earlier work.

c) Model theory: We provide a notion of model for parameterized algebraic theories (§VI). As we demonstrate, classical set theory is not a good way to understand parameterized algebraic theories, and so we develop model theory in a more general categorical setting. We then focus on models in functor categories which have been recognized as a good way to analyze local effects since the early 1980s (e.g. [43]).

d) Clones, monads and representations: In §VII we discuss abstract clones, which are presentation-invariant descriptions of terms-in-context modulo the equational theory. For example, the terms involving equality testing ($?_=$) can be enumerated by using Stirling numbers of the second kind. We use the abstract clones to produce an enriched monad, i.e. a notion of computation in the sense of Moggi [42].

e) Recovering earlier models of local store, restriction and π-calculus: Other researchers have investigated monads for local store ([49], [38]), restriction [47], and π-calculus [55]. We demonstrate the power of our framework by showing that our theories of local store, restriction and π respectively determine the local store monad, the restriction monad, and the monad for the π-calculus.

Our algebraic framework makes clear the design choices involved in this earlier monadic work, explaining the similarities and differences between these notions of computational effect. It is a convenient framework for exploring the variations and combinations of instances of computational effects.

II. Parameterized algebraic theories

We now introduce a variant of the framework of parameterized algebraic theories [59]. In §III–V we use this framework to describe instances of computational effects.

A. A type theory for parameterized computation

Our principal objective is to reason about effectful computation. We are not interested in the results of computations, rather in the effects that are occur during computation. However, a computation may depend on various parameters or indeed on another computation. We introduce a simple type system to analyze this. We assume a collection of base types, ranged over by α, β etc (informally, types of instances). We also have types such as $[\alpha, \beta]$ which is the type of a computation that depends on two instances, of type α and β respectively. We have the following general grammar of types:

$$\tau \;::=\; \alpha, \beta, \ldots \mid [\tau_1, \ldots, \tau_n]\ (n \geq 0)$$

so that a type is a tree with leaves labelled by base types. As usual, a context is a list of types annotated by distinct variables. There are two kinds of judgement: judgements of a computation in context ($\Gamma \vdash t$) and judgements of a typed expression in context ($\Gamma \vdash t : \tau$). The inhabitants of these judgements are defined inductively by the following five rules:

$$\frac{}{\Gamma, x\colon \alpha, \Gamma' \vdash x : \alpha}(\alpha \text{ is a base type}) \qquad \frac{\Gamma, x_1\colon \tau_1, \ldots, x_n\colon \tau_n \vdash t}{\Gamma \vdash x_1, \ldots, x_n.t : [\tau_1, \ldots, \tau_n]}(n > 0) \qquad \frac{\Gamma \vdash t}{\Gamma \vdash t : []}$$

$$\frac{\Gamma, x\colon [\tau_1 \ldots \tau_n], \Gamma' \vdash t_1 : \tau_1 \quad \ldots\ \Gamma, x\colon [\tau_1 \ldots \tau_n], \Gamma' \vdash t_n : \tau_n}{\Gamma, x\colon [\tau_1 \ldots \tau_n], \Gamma' \vdash x(t_1 \ldots t_n)}(n > 0) \qquad \frac{-}{\Gamma, x\colon [], \Gamma' \vdash x}$$

One can define a substitution operation: given $\Gamma, x\colon \tau \vdash t$ and $\Gamma \vdash u : \tau$, we can build $\Gamma \vdash t[u/x]$. (It is not trivial but quite standard; see e.g. [5].) Because of the assumption

that variables in a context are all different, substitution avoids implicit variable capture.

As a functional programming language, the type $[\tau_1, \dots, \tau_n]$ can be thought of as a function type into an unspecified return type (say, ρ): $\tau_1 \to \cdots \to \tau_n \to \rho$; the inhabitants are η-long β-normal forms. From the perspective of logic, our type $[\tau_1, \dots, \tau_n]$ could be written $\neg(\tau_1 \wedge \cdots \wedge \tau_n)$. This perspective suggests that we can simulate a general function type via continuation passing style (CPS) (see e.g. [33]):

$$(\tau_1 \to \cdots \to \tau_n \to \alpha) \stackrel{\text{def}}{=} [\tau_1, \dots, \tau_n, [\alpha]].$$

For illustration, let α be a base type of memory cells. We can think of the term $? : [[], \alpha, []], a{:}\alpha, x{:}[], y{:}[] \vdash ?(x, a, y)$ as a computation that reads location a; if it is 0, we continue as x, and if it is 1, we continue as y. We might also think of the term $\nu_0 : [[\alpha]], x : [\alpha] \vdash \nu_0(a.x(a))$ as a computation that allocates a new memory cell a and continues as x. We can form computations such as $\nu_0(a.?(x, a, y))$ which allocates a new memory cell and then reads it, continuing as either x or y.

One can investigate algebraic effects in the context of an elaborate type system (e.g. [27]) or indeed Haskell [22]. Here we use a restricted type system to emphasise the algebraic nature of the framework.

B. Parameterized algebraic theories

We now provide a system for axiomatizing equations between terms in the simple type theory. To maintain a connection with enriched monads (§VII) we focus on equations in contexts of order < 2. The order of a type is given by the height of its syntax tree: $\mathit{order}(\alpha) = 0$, and $\mathit{order}([\tau_1, \dots, \tau_n]) = 1 + \max\{\mathit{order}(\tau_1), \dots, \mathit{order}(\tau_n)\}$. We say that a context $(x_1 : \tau_1, \dots, x_n : \tau_n)$ is order i (resp. (i/j)) if the types τ_k are all exactly order i (resp. i or j).

Definition 1. Let $\alpha, \beta, \dots$ be base types. A *parameterized signature* Σ comprises a collection of function symbols equipped with order (1/2) types. The *terms over* Σ *in context* Γ are terms in context (Σ, Γ) (we will elide Σ when writing the context).

An *equation* over a signature Σ is an order (0/1) context Γ and two computation terms in the context (Σ, Γ), written $\Gamma \vdash t \equiv u$. A presentation of a *parameterized algebraic theory* comprises a parameterized signature Σ together with a collection of equations over Σ.

From a presentation we form a deduction system by closing the equality relation under reflexivity, symmetry, transitivity, congruence and substitution.

Similar notions of algebraic theory have been investigated elsewhere. In [59], I use this framework to describe predicate logic and the effects associated with logic programming. In [40], we use the same framework but from the perspective of 'generic effects'. The ideas originate from the work of Plotkin and Power ([50], [48]) who develop a model theory rather than an equational deduction system. Moving slightly further afield, this kind of syntactic formalism dates back to Aczel's formalism [2] which is the basis of the second-order equational logic of Fiore et al. ([16], [23]).

III. Classical algebraic theories

When there are no base types, a parameterized algebraic theory (Def. 1) is just an algebraic theory in the sense of universal algebra (e.g. [13]).

A. Algebraic effects associated with reading a binary digit

We begin by considering various notions of computation related to reading a binary digit of some form, whether from a network, from a memory cell, or from a random generator. We consider a function symbol $? : [[], []]$, written infix, with the understanding that $x \mathbin{?} y$ is the computation that reads a bit and proceeds either as x or as y depending on the result.

(We have not assumed a boolean type bool, but if we had done we would have a type isomorphism $[[], []] \cong [[\mathit{bool}]]$, so $?$ can be thought of as a CPS version of $\mathsf{read} : \mathit{unit} \to \mathit{bool}$.)

We consider the following equations that can be imposed:

$$u, v, x, y : [] \vdash (u \mathbin{?} v) \mathbin{?} (x \mathbin{?} y) \equiv (u \mathbin{?} x) \mathbin{?} (v \mathbin{?} y) \quad (?/?)$$
$$x : [] \vdash x \equiv x \mathbin{?} x \quad \text{(idem-?)}$$
$$u, v, x, y : [] \vdash (u \mathbin{?} v) \mathbin{?} (x \mathbin{?} y) \equiv u \mathbin{?} y \quad \text{(dup-?)}$$

The 'medial' law ($?/?$) says that the bits that we are reading come from a pool. Idempotence (idem-?) says that reading is not observable. The law (dup-?) implies ($?/?$): it says that there is only one digit in the pool, i.e., it is a memory cell. These kinds of equation provide a guide for reasoning about programs [22] and compiler optimizations [27].

1) Writing the bit: We can describe the effect of writing to a bit of memory by adding two unary function symbols $\mathrm{w}_0 : [[]]$, $\mathrm{w}_1 : [[]]$. The idea is that $\mathrm{w}_i(x)$ sets the bit to i and continues as x. It is subject to three equation schemes:

$$x : [] \vdash \mathrm{w}_i(\mathrm{w}_j(x)) \equiv \mathrm{w}_j(x) \quad (\mathrm{w}_i\mathrm{w}_j)$$
$$x_0, x_1 : [] \vdash \mathrm{w}_i(x_0 \mathbin{?} x_1) \equiv \mathrm{w}_i(x_i) \quad (\mathrm{w}_i?)$$
$$x : [] \vdash x \equiv \mathrm{w}_0(x) \mathbin{?} \mathrm{w}_1(x) \quad (?\mathrm{w})$$

Laws (idem-?) and (dup-?) follow from these three [37].

(We can think of w_i as a CPS version of a command $\mathsf{write}_i : \mathit{unit} \to \mathit{unit}$, and with a boolean type we could combine w_0 and w_1 to get $\mathsf{write} : \mathit{bool} \to \mathit{unit}$.)

2) Probability and non-determinism: We can describe random computation using a binary function symbol $\oplus : [[], []]$ that is medial, idempotent and symmetric: $x \oplus y \equiv y \oplus x$. The idea is that $x \oplus y$ is the computation that behaves as either x or y, with equal probability ([24], [1]). If $\oplus$ is also associative (e.g. has a unit) then it is a semilattice, and it describes non-deterministic rather than probabilistic choice.

When reading ($?$) is combined with choice $\oplus$, one usually assumes that the data is ready before the choice ('early', [45])

$$u, v, x, y : [] \vdash (u \mathbin{?} v) \oplus (x \mathbin{?} y) \equiv (u \mathbin{?} y) \oplus (x \mathbin{?} v) \quad \text{(early-?)}$$

IV. Parameterizing a classical theory

We now develop the idea of introducing many instances of a classical algebraic theory, using parameters. For example, to create a theory of reading many bits, one for each inhabitant of a parameter type α, we replace the read operation $? : [[], []]$

by a parameterized read operation $? : [[], \alpha, []]$ which we write infix as $x \,?_a\, y$. We parameterize the equations too:

$$a : \alpha, x : [] \vdash x \equiv x \,?_a\, x \qquad \text{(idem-}?_a\text{)}$$
$$a : \alpha, u, v, x, y : [] \vdash (u \,?_a\, v) \,?_a\, (x \,?_a\, y) \equiv u \,?_a\, y \qquad \text{(dup-}?_a\text{)}$$

We can now understand α as a type of memory cells, and the operation $?$ takes a parameter specifying which cell to read.

The reader can imagine how to apply the same process to any classical algebraic theory. There are often additional equations that, informally, specify that different instances of the theory do not affect each other. To make this precise we recall the idea of commutativity (e.g. [35],[15]).

A. Commutativity

Informally, two terms t and u are said to *commute* if it doesn't matter which one is executed first. We make this precise by defining a kind of sequential composition. If Γ and Δ are order 1 contexts, let $(\Gamma \times \Delta)$ be the order 1 context with a variable $xy : [\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_n]$ for every variable $x : [\vec{\alpha}^m]$ in Γ and every variable $y : [\vec{\beta}^n]$ in Δ. If Ξ is any context, and $\Xi, \Gamma \vdash t$ and $\Xi, \Delta \vdash u$, we define sequential composition $\Xi, \Gamma \times \Delta \vdash t \,;\, u$ by setting

$$x(\vec{t}^m) \,;\, y(\vec{u}^n) \stackrel{\text{def}}{=} \begin{cases} x(t_1;y(\vec{u}) \,, \ldots, \, t_m;y(\vec{u})) & \text{if } x \in \Xi \\ y(x(\vec{t});u_1 \,, \ldots, \, x(\vec{t});u_n) & \text{if } x \in \Gamma,\ y \in \Xi \\ xy(t_1, \ldots, t_m, u_1, \ldots, u_n) & \text{if } x \in \Gamma,\ y \in \Delta \end{cases}$$

where $a \,;\, u \stackrel{\text{def}}{=} a$ if $a : \alpha$, and $(\vec{x}.t) \,;\, u \stackrel{\text{def}}{=} \vec{x}.(t \,;\, u)$, and similarly $t \,;\, a \stackrel{\text{def}}{=} a$ if $a : \alpha$, and $t \,;\, (\vec{x}.u) \stackrel{\text{def}}{=} \vec{x}.(t \,;\, u)$.

Definition 2 (c.f. [35],[15])**.** Let Γ and Δ be order 1 contexts and let Ξ be a context. Two terms $\Xi, \Gamma \vdash t$ and $\Xi, \Delta \vdash u$ are said to commute (w.r.t. Ξ) if $\Xi, \Gamma \times \Delta \vdash t \,;\, u \equiv u \,;\, t$.

A parameterized algebraic theory is *commutative* if all terms commute. For example, the theory of reading a bit is commutative if we assume the medial law ($^?/_?$). But reading does not commute with writing: $\mathrm{w}_0(x \,?\, y) \equiv \mathrm{w}_0(x) \,?\, \mathrm{w}_0(y)$ is not derivable and indeed it is inconsistent.

B. Options for parameterization

When parameterizing a classical algebraic theory there are three options regarding additional commutativity laws:

1: No further equations: instances of the theory do not commute with each other. When reading from a network channel, different instances are different channels, and the order of reads is often important even across different channels.

2: If the starting theory is commutative then we can add equations saying that all instances of the theory commute with each other. For the effect of reading from memory, the different instances are different memory locations, and the results are not dependent on the order of access (an 'abide' law: [9, §4])

$$a, b : \alpha,\ u, v, x, y : [] \vdash (u \underset{b}{?} v) \underset{a}{?} (x \underset{b}{?} y) \equiv (u \underset{a}{?} x) \underset{b}{?} (v \underset{a}{?} y) \quad (^?_a/_?_b)$$

There is an instance where $a{=}b$, but that is the medial law.

3: We may say that *different* instances of the parameterized theory commute with each other using the theory of equality testing to be introduced in §IV-C. For a non-commutative theory we need to be careful. For example, the parameterized theory of writing a bit has two function symbols, $\mathrm{w}_0, \mathrm{w}_1 : [\alpha, []]$. The commutativity equation $a, b : \alpha,\ x : [] \vdash \mathrm{w}_0(a, \mathrm{w}_1(b, x)) \equiv \mathrm{w}_1(b, \mathrm{w}_0(a, x))$ is inappropriate because of the instance where $a = b$.

(We note that options (1)–(3) can be explained from a categorical perspective in terms of (1) coproducts of theories, (2) coproducts of commutative theories, and (3) a monoidal structure on theories, following Power [52].)

C. Equality testing

The theory of equality testing has one function symbol $?_= : [[], \alpha, \alpha, []]$ which takes two parameters and two arguments. It is written infix: $x \,?_{a=b}\, y$. There are four laws:

$$a, b : \alpha,\ x : [] \vdash x \equiv x \,?_{a=b}\, x \qquad \text{(idem-}?_=\text{)}$$
$$a, b, c, d : \alpha, u, v, x, y : [] \vdash (u \,?_{c=d}\, v) \,?_{a=b}\, (x \,?_{c=d}\, y) \equiv (u \,?_{a=b}\, x) \,?_{c=d}\, (v \,?_{a=b}\, y) \qquad (^?_=/_?_=)$$
$$a, b : \alpha,\ x : [\alpha], y : [] \vdash x(a) \,?_{a=b}\, y \equiv x(b) \,?_{a=b}\, y \qquad \text{(sub-}?_=\text{)}$$
$$a : \alpha,\ x, y : [] \vdash x \,?_{a=a}\, y \equiv x \qquad (?_=)$$

Each instance $?_{a=b}$ satisfies (dup-?). First, note that

$$(u \underset{a=b}{?} v) \underset{a=b}{?} x \overset{\text{(sub)}}{\equiv} (u \underset{a=a}{?} v) \underset{a=b}{?} x \overset{(?_=)}{\equiv} (u \underset{a=a}{?} y) \underset{a=b}{?} x \overset{\text{(sub)}}{\equiv} (u \underset{a=b}{?} y) \underset{a=b}{?} x$$

and (dup-?) follows from this:

$$(u \underset{a=b}{?} v) \underset{a=b}{?} (x \underset{a=b}{?} y) \equiv (u \underset{a=b}{?} y) \underset{a=b}{?} (x \underset{a=b}{?} y) \overset{(^?/_?)}{\equiv} (u \underset{a=b}{?} x) \underset{a=b}{?} (y \underset{a=b}{?} y) \equiv (u \underset{a=b}{?} u) \underset{a=b}{?} (y \underset{a=b}{?} y) \overset{\text{(idem)}}{\equiv} u \underset{a=b}{?} y.$$

Symmetry of equality ($x ?_{a=b} y \equiv x ?_{b=a} y$) can also be derived.

D. Combining equality testing with other theories

We can use the theory of equality testing to formalize a conditional equation "$a{\neq}b : \alpha, \Gamma \vdash t \equiv u$" by $a, b : \alpha,\ \Gamma \vdash t \equiv t \,?_{a=b}\, u$, provided we also ensure that equality testing commutes with all terms. In this way we can make precise option (3) in §IV-B, that different instances of the classical theory commute. If we also have an equation $a\colon \alpha,\ \Gamma \vdash t[^a/_b] \equiv t'$ then we can simplify the presentation by replacing the two equations by one: $a, b : \alpha, \Gamma \vdash t \equiv t' \,?_{a=b}\, u$.

For example, the parameterized theory of reading/writing memory has four function symbols $? : [[], \alpha, []]$, $\mathrm{w}_0 : [\alpha, []]$, $\mathrm{w}_1 : [\alpha, []]$, $?_= : [[], \alpha, \alpha, []]$, subject to the laws for equality (idem-$?_=$, $^?/_?_=$, $?_=$, sub-$?_=$), commutativity of reading ($^?_a/_?_b$) and parameterized versions of the laws for writing a bit which are combined with commutativity of different instances of reading and writing (we write $a{:=}i \,;\, x$ for $\mathrm{w}_i(a, x)$) as follows:

$$a,b{:}\alpha, x{:}[] \vdash a{:=}i \,;\, b{:=}j \,;\, x \equiv (a{:=}j \,;\, x) \underset{a=b}{?} (b{:=}j \,;\, a{:=}i \,;\, x) \quad (^W/_W)$$
$$a, b : \alpha,\ x_0, x_1 : [] \vdash a{:=}i \,;\, (x_0 \,?_b\, x_1) \equiv (a{:=}i \,;\, x_i) \,?_{a=b}\, ((a{:=}i \,;\, x_0) \,?_b\, (a{:=}i \,;\, x_1)) \qquad (^W/_?)$$
$$a : \alpha,\ x : [] \vdash x \equiv (a{:=}0 \,;\, x) \,?_a\, (a{:=}1 \,;\, x) \qquad (?W)$$

We also require equality to commute with all the terms:

$$a,b,c{:}\alpha, u,v,x,y{:}[] \vdash (u \underset{a=b}{?} v) \underset{c}{?} (x \underset{a=b}{?} y) \equiv (u \underset{c}{?} x) \underset{a=b}{?} (v \underset{c}{?} y) \quad (?/?_=)$$

$$a,b,c{:}\alpha, x,y{:}[] \vdash c{:=}i\,; (x \underset{a=b}{?} y) \equiv (c{:=}i\,; x) \underset{a=b}{?} (c{:=}i\,; y) \quad (W/?_=)$$

V. Arguments with parameters and restriction

A. Arguments with parameters

Instead of the operation $? : [[],[]]$ that reads a binary digit (§III-A), we can consider an operation $\mathrm{r} : [[\alpha]]$ that reads an *instance*. The idea is that $\mathrm{r}(a.x(a))$ first reads something of type α, binds it to a and continues as x. The laws (idem-?) and ($?/?$) in §III-A can be revisited in this setting:

$$x\colon [] \vdash x \equiv \mathrm{r}(a.x) \quad \text{(idem-r)}$$

$$x\colon [\alpha,\alpha] \vdash \mathrm{r}(a.\mathrm{r}(b.x(a,b))) \equiv \mathrm{r}(a.\mathrm{r}(b.x(b,a))) \quad (\mathrm{r}/\mathrm{r})$$

We can also have different instances of *this* theory, accommodated by two base types (α, β) and an operation $\mathrm{r} : [\beta,[\alpha]]$.

B. Restriction

We now consider a restriction operator $\nu : [[\alpha]]$. The idea is that $\nu(a.x(a))$ first generates an instance a and then continues as x (shorthand: $\nu a.x(a)$). We can think of restriction as reading an instance from a pool of instances; indeed we consider the following two equations (e.g. [39]):

$$x : [] \vdash \nu a.x \equiv x \quad \text{(idem-}\nu\text{)}$$

$$x : [\alpha,\alpha] \vdash \nu a.\nu b.x(a,b) \equiv \nu a.\nu b.x(b,a) \quad (\nu/\nu)$$

The restriction operator can be combined with other function symbols, and we require it to commute with them. For example, we combine the parameterized theory of reading a bit ($? : [[],\alpha,[]]$; idem-$?_a$, dup-$?_a$, $?_a/?_b$) with restriction ($\nu : [[\alpha]]$; idem-ν, ν/ν) using the following commutativity law:

$$a{:}\alpha,\, x,y{:}[\alpha] \vdash \nu b.(x(b) \underset{a}{?} y(b)) \equiv (\nu b.x(b)) \underset{a}{?} (\nu b.y(b)) \quad (\nu/?)$$

We thus build a theory of instantiating and reading bits.

C. Initialization

When we combine the theory of reading a bit with the theory of restriction, the meaning of $\nu a.(x\ ?_a\ y)$ is not determined. What is the initial value of the new bit a? We consider two possibilities: random initialization and specified initialization.

1) Random initialization: If we impose a symmetry axiom $x,y{:}[] \vdash \nu a.(x\ ?_a\ y) \equiv \nu a.(y\ ?_a\ x)$ then we can think of α as a type of random boolean variables, so that ν randomly initializes with 0 or 1 with equal probability. We can define a choice operator, $\oplus \stackrel{\text{def}}{=} x,y.\nu a.(x\ ?_a\ y) : [[],[]]$, as in §III-A2.

2) Specified initialization: The second option is to have two restriction operations $\nu_0 : [[\alpha]]$, $\nu_1 : [[\alpha]]$ that initialize the parameter with 0 and 1 respectively. They are subject to laws (idem-ν, ν/ν, $\nu/?$) and

$$x,y : [\alpha] \vdash \nu_0 a.(x(a)\ ?_a\ y(a)) \equiv \nu_0 a.x(a) \quad (\nu_0)$$

$$x,y : [\alpha] \vdash \nu_1 a.(x(a)\ ?_a\ y(a)) \equiv \nu_1 a.y(a) \quad (\nu_1)$$

We occasionally require that α is actually a boolean type:

$$a\colon \alpha,\, x\colon [\alpha] \vdash x(a) \equiv (\nu_0 b.x(b))\ ?_a\ (\nu_1 b.x(b)) \quad (\eta\text{-}?)$$

We can then think of ν_0 and ν_1 as CPS forms of the constants 0 and 1 respectively, and that $?$ is if-then-else (see §VI-D).

D. Combining restriction with equality

When we combine the theory of restriction with the theory of equality testing ($?_= : [[],\alpha,\alpha,[]]$; idem-$?_=$, $?_=/?_=$, $?_=$, sub-$?_=$), we ask that restriction commutes with equality testing but also that the new instantiation is indeed new:

$$a,b{:}\alpha, x,y{:}[\alpha] \vdash \nu c.(x(c) \underset{a=b}{?_=} y(c)) \equiv (\nu c.x(c)) \underset{a=b}{?_=} (\nu c.y(c)) (\nu/?_=)$$

$$a{:}\alpha,\, x,y{:}[\alpha] \vdash \nu b.(x(b) \underset{a=b}{?_=} y(b)) \equiv \nu b.y(b) \quad \text{(new-}\nu\text{)}$$

E. Example: local store

We achieve a full theory of local store by carefully combining the theories of reading and writing a bit of memory, equality testing, and restriction. The function symbols are

$$? : [[],\alpha,[]] \quad \mathrm{w}_0,\mathrm{w}_1 : [\alpha,[]] \quad ?_= : [[],\alpha,\alpha,[]] \quad \nu_0,\nu_1 : [[\alpha]].$$

We impose the laws for reading and writing a bit (W/W, $W/?$, $?W$), for equality (idem-$?_=$, $?_=/?_=$, $?_=$, sub-$?_=$), for restriction (idem-ν, ν/ν), the commutativity laws ($?/?_=$, $W/?_=$, $\nu/?$, $\nu/?_=$), the interaction laws (ν_0, ν_1, new-ν) and two additional schemes:

$$a\colon \alpha,\, x\colon [\alpha] \vdash \mathrm{w}_i(a, \nu_j b.x(b)) \equiv \nu_j b.\mathrm{w}_i(a, x(b)) \quad (\mathrm{w}_i/\nu_j)$$

$$x\colon [\alpha] \vdash \nu_i a.\mathrm{w}_j(a, x(a)) \equiv \nu_j a.x(a) \quad (\nu\mathrm{w})$$

We do not include (η-?) since it is inconsistent with (new-ν).

F. Example: algebraic presentation of the π-calculus

Milner's π-calculus [39] is a system that describes sending and receiving messages on named channels. In its simplest form, the only messages that can be sent are the channel names themselves. Thus we are in the setting of §V-A but with only one base type $\alpha = \beta$, which is a type of channel names. We can equip our computations with the primitives of the π-calculus since the axiomatization of early congruence ([45],[57]) can be made formal using the framework of parameterized algebraic theories. We consider seven function symbols: equality testing ($?_= : [[],\alpha,\alpha,[]]$), restriction ($\nu : [[\alpha]]$), choice ($\oplus : [[],[]]$), deadlock ($\bot$), output prefix ($\mathrm{Out} : [\alpha,\alpha,[]]$, shorthand $\bar{a}b.x$), input prefix ($\mathrm{In} : [\alpha,[\alpha]]$, shorthand $a(b).x(b)$), and silent prefix ($\mathrm{Tau} : [[]]$).

We have the laws of equality testing and restriction (idem-$?_=$, $?_=/?_=$, $?_=$, sub-$?_=$, idem-ν, ν/ν, $\nu/?_=$, new-ν); laws saying that $\oplus$ is medial, idempotent and symmetric; the unit law $\bot \oplus x \equiv x$; laws asserting that ν and $?_=$ commute with everything; and the following laws for early input (c.f. §III-A2) and laws asserting that communication on a hidden channel is unobservable:

$$a,b{:}\alpha, u,x{:}[], v,y{:}[\alpha] \vdash a(c).(u\ ?_{b=c}\ v(c)) \oplus a(c).(x\ ?_{b=c}\ y(c)) \equiv a(c).(u\ ?_{b=c}\ y(c)) \oplus a(c).(x\ ?_{b=c}\ v(c))$$

$$b{:}\alpha,\, x{:}[\alpha] \vdash \nu a.\bar{a}b.x(a) \equiv \bot \quad x{:}[\alpha] \vdash \nu a.\bar{a}a.x(a) \equiv \bot$$

$$x{:}[\alpha,\alpha] \vdash \nu a.a(b).x(a,b) \equiv \bot$$

VI. MODELS

Model theory is a powerful tool for investigating theories. It is widely recognized that classical set theory is not a perfect place to model syntax with binding – a point that we illustrate in §VI-B–C. For this reason we jump to a notion of model that works in many categories. We find a complete notion of model through interpretations in a functor category (§VI-E).

A. Models in categories: definition and examples

Recall that a category $\mathcal{C}$ is *cartesian closed* if it has finite products and each hom-functor $\mathcal{C}(- \times A, B) : \mathcal{C}^{\mathrm{op}} \to \mathcal{S}et$ is representable: there is an object B^A and a natural isomorphism $\mathcal{C}(C, B^A) \cong \mathcal{C}(C \times A, B)$ ("currying", e.g. [7, §5.4]). The category of sets and functions is cartesian closed: categorical products are cartesian products of sets, and the representation of $\mathcal{S}et(- \times A, B)$ is the set of functions B^A.

Consider a cartesian closed category $\mathcal{C}$ with a chosen object $[\![\alpha]\!]$ for each base type α. Given an object X of $\mathcal{C}$ we interpret a type τ (as in §II-A) as an object $[\![\tau]\!]_X$, by induction on the structure of types: let $[\![\alpha]\!]_X \stackrel{\text{def}}{=} [\![\alpha]\!]$ and let $[\![[\tau_1 \dots \tau_n]]\!]_X \stackrel{\text{def}}{=} X^{([\![\tau_1]\!]_X \times \cdots \times [\![\tau_n]\!]_X)}$. We also interpret contexts $(x_1{:}\tau_1 \dots x_n{:}\tau_n)$ as objects $[\![\tau_1]\!]_X \times \cdots \times [\![\tau_n]\!]_X$.

Definition 3. Let Σ be a parameterized signature (Def. 1). A Σ-*structure* in a cartesian closed category $\mathcal{C}$ is given by an object $\mathfrak{M}$ and a morphism $[\![F]\!]_{\mathfrak{M}} : [\![\tau_1]\!]_{\mathfrak{M}} \times \cdots \times [\![\tau_n]\!]_{\mathfrak{M}} \to \mathfrak{M}$ for each symbol $F : [\tau_1, \dots, \tau_n]$ in Σ. A *homomorphism* between Σ-structures $\mathfrak{M}$, $\mathfrak{N}$ in $\mathcal{C}$ is a morphism $f : \mathfrak{M} \to \mathfrak{N}$ in $\mathcal{C}$ that commutes with the interpretations of Σ.

For example, a structure for equality testing has a morphism $[\![?_=]\!] : \mathfrak{M} \times [\![\alpha]\!] \times [\![\alpha]\!] \times \mathfrak{M} \to \mathfrak{M}$, and a structure for restriction has a morphism $[\![\nu]\!] : \mathfrak{M}^{[\![\alpha]\!]} \to \mathfrak{M}$.

In a structure we can interpret a judgement $\Gamma \vdash t$ as a morphism $[\![t]\!]_{\mathfrak{M}} : [\![\Gamma]\!]_{\mathfrak{M}} \to \mathfrak{M}$. This interpretation is defined straightforwardly by induction on derivations.

Definition 4. A structure $\mathfrak{M}$ satisfies an equation $\Gamma \vdash t \equiv u$ if the morphisms $[\![t]\!]_{\mathfrak{M}}, [\![u]\!]_{\mathfrak{M}} : [\![\Gamma]\!]_{\mathfrak{M}} \to \mathfrak{M}$ are equal. A model of a theory is a structure in which all equations are satisfied.

Examples: If $\mathcal{C} = \mathcal{S}et$ and there are no base types then a model is the same thing as a model in the classical sense [13]: a set $\mathfrak{M}$ together with an n-ary function $\mathfrak{M}^n \to \mathfrak{M}$ for each n-ary function symbol, satisfying the equations. In §VI-E we investigate models in functor categories. Before that, we look briefly at set-theoretic models in more detail.

B. Set-theoretic models of equality testing

Proposition 5. *For any sets $[\![\alpha]\!]$ and $\mathfrak{M}$ there is a unique function $[\![?_=]\!] : \mathfrak{M} \times [\![\alpha]\!] \times [\![\alpha]\!] \times \mathfrak{M} \to \mathfrak{M}$ making $\mathfrak{M}$ a model of the theory of equality testing (§IV-C) in the category of sets.*

Proof: Given sets $[\![\alpha]\!]$ and $\mathfrak{M}$ we have a model satisfying $[\![?_=]\!]_{\mathfrak{M}}(x, a, a, y) = x$ and $[\![?_=]\!]_{\mathfrak{M}}(x, a, b, y) = y$ for $a \neq b$. In fact, all models are of this form. Consider a set $\mathfrak{M}$ with a function $?_= : \mathfrak{M} \times [\![\alpha]\!] \times [\![\alpha]\!] \times \mathfrak{M} \to \mathfrak{M}$ satisfying the axioms of equality testing. For all $a, b \in [\![\alpha]\!]$, either $a = b$ or $a \neq b$. If $a = b$ then $x \mathbin{?_{a=b}} y = x$ by axiom ($?_=$). The interesting case is where $a \neq b$. We have a function $z : [\![\alpha]\!] \to \mathfrak{M}$ satisfying $z(a) = x$ and $z(c) = y$ if $a \neq c$, so that

$$x \mathbin{?_{a=b}} y = z(a) \mathbin{?_{a=b}} y \overset{\text{(sub)}}{=} z(b) \mathbin{?_{a=b}} y = y \mathbin{?_{a=b}} y \overset{\text{(idem)}}{=} y. \qquad \blacksquare$$

C. No set-theoretic models of restriction

Proposition 6. *Let $[\![\alpha]\!]$ be a countable set. There are no non-trivial set-theoretic models of restriction and equality (§V-D).*

Proof: We consider a model $\mathfrak{M}$ with two arbitrary elements, x and y, and show that $x = y$. By Prop. 5, $[\![?_=]\!]$ is uniquely determined. We have the following consequence of axiom (new-ν): for any $f : [\![\alpha]\!] \to \mathfrak{M}$,

$$\text{if } \{a \in [\![\alpha]\!] \mid f(a) = x\} \text{ is cofinite then } \nu(f) = x. \qquad (2)$$

Since $[\![\alpha]\!]$ is countable, there is a total order $\leq$ on it such that every down-set is finite. We define a function $f_{x,y} : [\![\alpha]\!] \times [\![\alpha]\!] \to \mathfrak{M}$ such that $f_{x,y}(a, b) = x$ if $a \leq b$ and $f_{x,y}(a, b) = y$ if $a > b$. Note that for fixed a, $f_{x,y}(a, -)$ is cofinitely x, and for fixed b, $f_{x,y}(-, b)$ is cofinitely y. Thus

$$\begin{aligned} x &\overset{\text{(idem)}}{=} \nu(\lambda a.x) \overset{(2)}{=} \nu(\lambda a.\nu(\lambda b.f_{x,y}(a, b))) \\ &\overset{(\nu/\nu)}{=} \nu(\lambda b.\nu(\lambda a.f_{x,y}(a, b))) \overset{(2)}{=} \nu(\lambda b.y) \overset{\text{(idem)}}{=} y. \end{aligned} \qquad \blacksquare$$

In Section VI-E we consider a functor category in which there is an object $[\![\alpha]\!]$ supporting non-trivial models of restriction (see also Prop. 13; Thm. 15). For another example, the atoms of the FM model of set theory support non-trivial models of restriction (e.g. [47], [20]). It remains to be seen whether there is an uncountable set $[\![\alpha]\!]$ in classical ZFC that supports non-trivial models of restriction.

D. Models of specified initialization with η

In any category, the theory of restriction with specified initialization and η (§V-C2) can be seen as a theory that asserts that the parameter type α is a type of booleans $(1 + 1)$.

Proposition 7. *Consider the theory of specified initialization with η (§V-C2). Let $\mathcal{C}$ be a cartesian closed category with a specified object $[\![\alpha]\!]$. The following are equivalent:*

1) *Every object of $\mathcal{C}$ can be assigned a model structure in such a way that every morphism is a homomorphism.*
2) *The object $[\![\alpha]\!]$ is a coproduct of the terminal object with itself ($[\![\alpha]\!] = (1 + 1)$).*

E. Models in a category of functors

1) Category of contexts: We begin by defining a category $\mathcal{C}tx_0$. The objects are order 0 contexts. A morphism $(a_1 \colon \alpha_1, \dots, a_p \colon \alpha_p) \to (b_n \colon \beta_1, \dots, b_q \colon \beta_q)$ is a renaming, i.e. a function $f : p \to q$ such that $\alpha_i = \beta_{f(i)}$ $(1 \leq i \leq p)$.

2) Presheaf category: We form the category $[\mathcal{C}tx_0, \mathcal{S}et]$ of functors $\mathcal{C}tx_0 \to \mathcal{S}et$. The objects of this category are functors $X : \mathcal{C}tx_0 \to \mathcal{S}et$, i.e., for each order 0 context Γ a set $X(\Gamma)$ is given, and for each renaming $f : \Gamma \to \Gamma'$, a function $Xf : X(\Gamma) \to X(\Gamma')$ is given, such that identities and composition are respected. The morphisms between functors are natural transformations, i.e. families of functions

$\{\phi_\Gamma : X(\Gamma) \to Y(\Gamma)\}_\Gamma$ that respect the renamings. We call the functors $\mathcal{C}tx_0 \to \mathcal{S}et$ 'presheaves'.

For example, consider an order 1 context Δ and let $Terms(\Delta|\Gamma)$ be the set of terms in context (Δ, Γ) modulo the equations. With the evident renaming structure, this yields a presheaf $Terms(\Delta|-) : \mathcal{C}tx_0 \to \mathcal{S}et$.

3) Base types: Let $[\![\alpha]\!]$ be the representable presheaf $[\![\alpha]\!](\Gamma) = \mathcal{C}tx_0((\alpha), \Gamma)$.

4) Cartesian closed structure: The functor category $[\mathcal{C}tx_0, \mathcal{S}et]$ has products given pointwise: $(X \times Y)(p) = X(p) \times Y(p)$. The cartesian closed structure satisfies $Y^X(\Gamma) \cong [\mathcal{C}tx_0, \mathcal{S}et]([\![\Gamma]\!] \times X, Y)$. In particular $Y^{[\![\Gamma]\!]}(\Gamma') = Y(\Gamma, \Gamma')$.

5) Models: We can now study models in $[\mathcal{C}tx_0, \mathcal{S}et]$. An important first result is that for any order 1 context Δ we have a term model $\mathfrak{M} = Terms(\Delta|-) : [\mathcal{C}tx_0, \mathcal{S}et]$. For each function symbol $F : [\tau_1, \ldots, \tau_n]$ in Σ, we have a morphism $[\![F]\!]_\mathfrak{M} : [\![\tau_1]\!]_\mathfrak{M} \times \cdots \times [\![\tau_n]\!]_\mathfrak{M} \to \mathfrak{M}$ given by term formation.

Proposition 8 (Completeness)**.** *Let Γ be an order 0 context and let Δ be an order 1 context. An equation $\Gamma, \Delta \vdash t \equiv u$ is satisfied in the model $Terms(\Delta|-)$ (in $[\mathcal{C}tx_0, \mathcal{S}et]$) if and only if it is derivable in the theory.*

VII. Representations of theories

Two different presentations, with different function symbols or equations, may induce the same notion of model. In classical universal algebra, *abstract clones* provide a presentation-invariant description of theories [13, Ch. III]. We now discuss a theory of abstract clones for parameterized algebraic theories.

Roughly speaking, an abstract clone is given by the set of terms modulo equations. Often there is a non-syntactic description of the terms modulo equations. As a starting point we recall a result for the classical theory for reading a bit of memory (§III-A: idem-?, dup-?). Every term can be uniquely rewritten to one of the form $x\ ?\ y$, so there are n^2 terms in n variables. In the theory of reading/writing memory (§III-A1) every term can be uniquely rewritten to the form $\mathrm{w}_i(x)\ ?\ \mathrm{w}_j(y)$ for $i, j \in \{0, 1\}$, so there are $4n^2$ terms in n variables ([37],[49]). We characterize equality testing and restriction in a similar way (§VII-C–E).

Abstract clones are well-known to be equivalent to Lawvere theories and finitary monads. Other authors have proposed *indexed Lawvere theories* (Power, [51]) and *graded Lawvere theories* (Melliès, [38]) to study instances of effects. The development in this section takes inspiration from those ideas but is tailored to (a) a precise connection with parameterized algebraic theories and (b) enrichment in a cartesian closed structure.

A. Enriched abstract clones and monads

Definition 9 ([59], [30])**.** Let $\mathcal{C}$ be a cartesian closed category. Let $\mathbb{F}$ be a dense full subcategory. An *abstract clone* is given by

- for each object A of $\mathbb{F}$ an object TA of $\mathcal{C}$;
- a morphism $\eta : A \to TA$ for each object A of $\mathbb{F}$;
- a morphism $\gg= : TA \times (TB)^A \to TB$ for objects A, B of $\mathbb{F}$

such that the following equations hold in the internal language of cartesian closed categories, where we write $\gg=$ infix:

$$x \gg= \eta \;=\; x \qquad (\eta x) \gg= f \;=\; f(x)$$
$$(x \gg= f) \gg= g \;=\; x \gg= (\lambda a.\, (f(a) \gg= g)).$$

When $\mathbb{F}$ comprises all the objects of $\mathcal{C}$, an abstract clone is simply an enriched monad (i.e. a strong monad, e.g. [42]).

When $\mathcal{C}$ is the category of sets and $\mathbb{F}$ comprises the natural numbers considered as sets, then we have the original definition of abstract clone, attributed to P Hall [13, Ch. III]. Any classical algebraic theory induces an abstract clone, where TA is the set of all A-ary derived operations, or equivalently the set of all terms in a context with A variables, modulo the equations; every abstract clone arises in this way. Classical abstract clones have been generalized to Kleisli structures (e.g. [14]) and relative monads ([6], [61]).

When $\mathcal{C}$ is locally small and cocomplete then we can left-Kan-extend any abstract clone to a strong monad. Since $\mathbb{F}$ is dense in $\mathcal{C}$, every object X of $\mathcal{C}$ is a colimit $X = W \star D$ of a canonical diagram $D : J \to \mathbb{F}$ of objects of $\mathbb{F}$, relative to a weight $W : J^{\mathrm{op}} \to \mathcal{C}$ (e.g. [28]). We define an enriched monad on $\mathcal{C}$ by preserving this colimit: $T(W \star D) \stackrel{\text{def}}{=} W \star (T(D-))$. More concretely, $T(X)$ is a coequalizer:

$$\coprod_{A,B \in \mathbb{F}} T(A) \times B^A \times X^B \rightrightarrows \coprod_{A \in \mathbb{F}} T(A) \times X^A \to T(X)$$

whose components can be thought of as pairs of a term together with a valuation for its context, modulo change of variables. See also [8], [30]: monads that arise in this way are an enriched version of 'monads with arities'.

Aside: From an abstract clone T for $(\mathbb{F} \subseteq \mathcal{C})$ we can build a $\mathcal{C}$-enriched category $\mathcal{C}_T$: the objects are the objects of $\mathbb{F}$, and the hom-objects are given by $\mathcal{C}_T(A, B) \stackrel{\text{def}}{=} [B \Rightarrow T(A)]$. This is essentially an enriched Lawvere theory [53] (albeit with different objects, following [6], [8], [11], [32], [37], [38], [51]), or dually a distributive Freyd category ([34], [40]).

B. Clones on presheaf categories

We have already seen (§VI-E5) that the presheaf category $[\mathcal{C}tx_0, \mathcal{S}et]$ is a good place to model parameterized algebraic theories. For the subcategory $\mathbb{F}$, we choose the category of order 1 contexts.

To describe an order 1 context it is sufficient to describe the number of variables of type $[\alpha_1 \ldots \alpha_m]$, for each list of order 0 types, $\alpha_1 \ldots \alpha_m$. We can thus understand an order 1 context $\Gamma = (x_1 \colon [\alpha_{1,1}, \ldots, \alpha_{1,m_1}], \ldots x_n \colon [\alpha_{1,1}, \ldots, \alpha_{n,m_n}])$ as a sum of representable presheaves:

$$\Gamma = \mathcal{C}tx_0((\alpha_{1,1} \ldots \alpha_{1,m_1}), -) + \cdots + \mathcal{C}tx_0((\alpha_{n,1} \ldots \alpha_{n,m_n}), -)$$

so that the interpretation of a context given in §VI-A satisfies $[\![\Gamma]\!]_X \cong X^\Gamma$.

Recall that the category of finite sums of representables $\mathcal{C}tx_1$ is a finite coproduct completion of $\mathcal{C}tx_0$ (e.g. [10]). The importance of this construction to algebra is well established (e.g. [26] and §VIII-B3).

The term model construction (§VI-E5) gives the following correspondence:

Theorem 10 ([59])**.** *To give a parameterized algebraic theory is to give an abstract clone for $\mathcal{C}tx_1 \subseteq [\mathcal{C}tx_0, \mathcal{S}et]$.*

C. Example: clone for equality testing

We now provide a syntax free description of the abstract clone for the theory of equality testing (§IV-C). Since there is only one base type in that theory, an order 0 context is given by a natural number (its size) and an order 1 context is given by a function $k\colon \mathbb{N} \to \mathbb{N}$ such that the set $\{n \in \mathbb{N} \mid k(n) \neq 0\}$ is finite, so that $k(n)$ is the number of variables of type $[\alpha^n]$. So an order (0/1) context is given by a pair $(k|p)$.

Recall that the Stirling number of the second kind, $S_p(n)$, is the number of ways to partition a set of size p into n non-empty subsets (e.g. [54, §1.9]). The function $S_p : \mathbb{N} \to \mathbb{N}$ has finite support, and so we can understand it as an order 1 context.

Proposition 11. *The terms of the theory of equality testing (§IV-C) in context $(k|p)$ are in bijection with the set of renamings $\mathcal{C}tx_1(S_p, k)$.*

Before embarking on a proof we note the general fact that we can count the morphisms of order 1 contexts, $k \to l$:

$$\mathcal{C}tx_1(k,l) \;\cong\; \textstyle\prod_m (\sum_n (l(n) \times m^n))^{k(m)} \tag{3}$$

since a morphism assigns to each variable of type $[\alpha^m]$ in k a variable of type $[\alpha^n]$ in l together with a renaming of the parameter variables, $n \to m$.

Proof notes for Prop. 11: We convert a term into a normal form as follows. Construct a binary tree of height $p \times p$ where the nodes at height (i,j) are labelled $?_{a_i = a_j}$. By law (idem-?), any term t is equal to this tree with t at each of the leaves. We now use ($?_{=}/?_{=}$) and the derivable law (dup-?) to reduce the tree to one where the leaves are of the form $x(\vec{a})$. Each of the $2^{p \times p}$ leaves can be tagged with a binary relation on p which contains (i,j) if the path to the leaf turned left at height (i,j). If this binary relation is not an equivalence relation, then the leaf doesn't matter, by ($?_{=}/?_{=}$, $?_{=}$, sub-$?_{=}$, dup-?). If it is an equivalence relation then we can rename the variables $\vec{a}$ within the equivalence relation. Thus for each equivalence relation on p with m partitions a variable $x : n$ must be given together with parameters, of which there are m^n. ∎

D. Example: clone for restriction

We enumerate the partial equivalence relations on a set of size n whose domain is size d: let $\mathrm{PER}(n,d) \stackrel{\text{def}}{=} \binom{n}{d} \times B_d$, where $\binom{n}{d}$ is the number of ways to choose d from n (the binomial coefficient), and $B_d \stackrel{\text{def}}{=} \sum_m S_d(m)$ is the number of equivalence relations on d (the Bell number, [54, §1.9]). We now use this to define

$$(\mathrm{Res}\, k)(n) \stackrel{\text{def}}{=} \textstyle\sum_d k(n+d) \times \mathrm{PER}(n+d, d) \tag{4}$$

Let $\delta_p\colon \mathbb{N} \to \mathbb{N}$ satisfy $\delta_p(p) = 1$ and $\delta_p(n) = 0$ $(n \neq p)$.

Proposition 12. *The terms of the theory of restriction (§V-B) in context $(k|p)$ are in bijection with the set of renamings $\mathcal{C}tx_1(\delta_p, \mathrm{Res}\, k)$.*

Proof notes: Given a renaming $\phi : \mathcal{C}tx_1(\delta_p, \mathrm{Res}\, k)$ we let $(n, (d, i, (s, (m, q))), f) = \phi(p)(*)$, via (3) and (4). So x_i is a variable of type $[\alpha^{n+d}]$, s is a subset of $(n+d)$ of size d, q is partition of d into m sets, and $f\colon n \to p$ is a function. We build the term $\nu b_1 \ldots \nu b_m . x_i(c_1, \ldots, c_{n+d})$ where $c_j = a_{f(j')}$ if j is the j'-th element of $(n+d) \setminus s$, and $c_j = b_{g(j')}$ if j is the j'-th element of s, and $g\colon d \to m$ is a choice of surjection that implements q. ∎

E. Example: the abstract clone for restriction and equality

Proposition 13. *The terms of the theory of restriction and equality (§V-D) in context $(k|p)$ are in bijection with the set of renamings $\mathcal{C}tx_1(S_p, \mathrm{Res}\, k)$.*

These representation results can be used as a starting point for representation results for other theories, including π-calculus and local store.

VIII. Relating theories

In this section we assume that there is only one base type, so that an order 0 context is determined by a natural number. Thus the category $\mathcal{C}tx_0$ comprises natural numbers and all functions between them.

Many authors have considered the subcategory $\mathcal{I}nj_0$ of natural numbers and injective functions. We will consider three categories of algebras over $[\mathcal{I}nj_0, \mathcal{S}et]$ from the literature:

1) algebras for local store [49],
2) algebras for restriction [47],
3) algebras for pi calculus [56].

We will explain the connection between these algebras and the parameterized algebraic theories for these notions of computation.

A. Main results

First, we recall that the identity-on-objects inclusion functor $J : \mathcal{I}nj_0 \to \mathcal{C}tx_0$ induces a functor $J^* : [\mathcal{C}tx_0, \mathcal{S}et] \to [\mathcal{I}nj_0, \mathcal{S}et]$ (by precomposition) which has a right adjoint $J_* : [\mathcal{I}nj_0, \mathcal{S}et] \to [\mathcal{C}tx_0, \mathcal{S}et]$. (See also [19].)

1) Local store: Plotkin and Power defined a monad for local store by defining a category $LS\text{-}alg$ of 'local store algebras', with a forgetful functor $LS\text{-}alg \to [\mathcal{I}nj_0, \mathcal{S}et]$ which they showed was monadic (i.e., $LS\text{-}alg$ is the category of Eilenberg-Moore algebras for a monad).

Theorem 14. *Let $\mathcal{M}od(LS; [\mathcal{C}tx_0, \mathcal{S}et])$ be the category of models and homomorphisms for the parameterized algebraic theory of local store (§V-E) in the category $[\mathcal{C}tx_0, \mathcal{S}et]$. There is an equivalence of categories $LS\text{-}alg \simeq \mathcal{M}od(LS; [\mathcal{C}tx_0, \mathcal{S}et])$ making the following diagram commute.*

$$\begin{array}{ccc} LS\text{-}alg & \simeq & \mathcal{M}od(LS; [\mathcal{C}tx_0, \mathcal{S}et]) \\ \downarrow & & \downarrow \\ [\mathcal{I}nj_0, \mathcal{S}et] & \xrightarrow[J_*]{} & [\mathcal{C}tx_0, \mathcal{S}et] \end{array} \tag{5}$$

In fact, Plotkin and Power showed that their functor $LS\text{-}alg \to [\mathcal{I}nj_0, \mathcal{S}et]$ is a $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched functor between $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched categories. Since the right adjoint $J_* : [\mathcal{I}nj_0, \mathcal{S}et] \to [\mathcal{C}tx_0, \mathcal{S}et]$ preserves products it yields a

2-functor $[\mathcal{I}nj_0, \mathcal{S}et]$-$\mathbf{CAT} \to [\mathcal{C}tx_0, \mathcal{S}et]$-$\mathbf{CAT}$. Indeed, we can understand the commuting diagram (5) as a diagram in the 2-category of $[\mathcal{C}tx_0, \mathcal{S}et]$-enriched categories and functors.

2) Restriction: Pitts [47] and others (e.g. [20], [36]) have used 'restriction structures' to model various effects relating to name generation and restriction. This category is monadic over $[\mathcal{I}nj_0, \mathcal{S}et]$, and the corresponding dynamic allocation monad has been used to model the ν-calculus [56, §5].

Theorem 15. *Let $\mathcal{M}od(\nu, ?_=; [\mathcal{C}tx_0, \mathcal{S}et])$ be the category of models of restriction with equality testing (§V-D) in $[\mathcal{C}tx_0, \mathcal{S}et]$. There is an equivalence of categories $\mathcal{M}od(\nu, ?_=; [\mathcal{C}tx_0, \mathcal{S}et]) \simeq \mathcal{R}es$ making the following diagram commute:*

$$\begin{array}{ccc} \mathcal{R}es & \simeq & \mathcal{M}od(\nu, ?_=; [\mathcal{C}tx_0, \mathcal{S}et]) \\ \downarrow & & \downarrow \\ [\mathcal{I}nj_0, \mathcal{S}et] & \xrightarrow[J_*]{} & [\mathcal{C}tx_0, \mathcal{S}et] \end{array}$$

3) π-calculus: Stark [57, §5.2] introduced a category $\mathcal{PI}$ of 'algebras' with a monadic forgetful functor $\mathcal{PI} \to [\mathcal{I}nj_0, \mathcal{S}et]$. The monad over $[\mathcal{I}nj_0, \mathcal{S}et]$ describes the π-calculus up-to early bisimulation [57, eq. (4)].

Theorem 16. *Let $\mathcal{M}od(\pi; [\mathcal{C}tx_0, \mathcal{S}et])$ be the category of models of the theory of the π-calculus (§V-F) in $[\mathcal{C}tx_0, \mathcal{S}et]$. There is an equivalence of categories $\mathcal{M}od(\pi; [\mathcal{C}tx_0, \mathcal{S}et]) \simeq \mathcal{PI}$ making the following diagram commute:*

$$\begin{array}{ccc} \mathcal{PI} & \simeq & \mathcal{M}od(\pi; [\mathcal{C}tx_0, \mathcal{S}et]) \\ \downarrow & & \downarrow \\ [\mathcal{I}nj_0, \mathcal{S}et] & \xrightarrow[J_*]{} & [\mathcal{C}tx_0, \mathcal{S}et] \end{array}$$

This theorem substantiates a proposal in [57, §6]. Notice that by composing with the adjunction $J^* \dashv J_*$, early bisimulation becomes early congruence.

B. Details on the general situation

We now explain the general connection between algebras and monads on $[\mathcal{I}nj_0, \mathcal{S}et]$ and parameterized algebraic theories, to suggest how Theorems 14–16 are proved.

1) Characterization of monads: In Section VII-B we showed that parameterized algebraic theories can be understood as abstract clones in $(\mathcal{C}tx_1 \subseteq [\mathcal{C}tx_0, \mathcal{S}et])$. We have the following general result about the monads that correspond to abstract clones on presheaf categories.

Proposition 17 ([30],[32],[59])**.** *Let $\mathbb{C}$ be a small category and let $\mathbb{F}$ be the full subcategory of $[\mathbb{C}, \mathcal{S}et]$ comprising the finite coproducts of representables. The following data are equivalent:*

1) *An abstract clone for $\mathbb{F} \subseteq [\mathbb{C}, \mathcal{S}et]$.*
2) *A strong monad on $[\mathbb{C}, \mathcal{S}et]$ that preserves filtered colimits and reflexive coequalizers.*
3) *A $[\mathbb{C}, \mathcal{S}et]$-enriched category $\mathcal{A}$ with filtered colimits, reflexive coequalizers and powers, together with a functor $\mathcal{A} \to [\mathbb{C}, \mathcal{S}et]$ that preserves filtered colimits, reflexive coequalizers and powers, reflects isomorphisms, and has a left adjoint.*

Proof notes: 1↔2: by left Kan extension, as in §VII-A. The density presentation of $\mathbb{F} \subseteq [\mathbb{C}, \mathcal{S}et]$ can be chosen to comprise the filtered colimits and reflexive coequalizers (see also [29, Thm. 7.2]). Indeed, $[\mathbb{C}, \mathcal{S}et]$ is the free completion of $\mathbb{F}$ under filtered colimits and reflexive coequalizers ([3], [4], [32]). 2↔3: an enriched version of the crude monadicity theorem. ∎

We define a *$[\mathbb{C}, \mathcal{S}et]$-enriched finitary crude-monadic functor* to be a functor as in item (3) of Prop. 17.

2) Connections between notions of algebra: In general we have a connection between certain monads T on $[\mathcal{I}nj_0, \mathcal{S}et]$ and parameterized algebraic theories.

Definition 18. We say that a presheaf X in $[\mathcal{I}nj_0, \mathcal{S}et]$ is 1-*inhabited* if either $X(n)$ is empty for all n, or $X(1)$ is inhabited. A functor $F : \mathcal{A} \to [\mathcal{I}nj_0, \mathcal{S}et]$ is 1-*inhabited* if $F(A)$ is 1-inhabited for all A.

The functors from earlier work ($\mathcal{R}es \to [\mathcal{I}nj_0, \mathcal{S}et]$ [47], $LS\text{-}alg \to [\mathcal{I}nj_0, \mathcal{S}et]$ [49], $\mathcal{PI} \to [\mathcal{I}nj_0, \mathcal{S}et]$ [57]) are all $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched finitary crude-monadic and 1-inhabited. This allows us to relate them with parameterized algebraic theories.

Theorem 19. *Consider a $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched finitary crude-monadic functor $\mathcal{A} \to [\mathcal{I}nj_0, \mathcal{S}et]$ that is 1-inhabited. There is a parameterized algebraic theory $\mathbb{T}$ yielding the following situation of $[\mathcal{C}tx_0, \mathcal{S}et]$-enriched categories and functors.*

$$\begin{array}{ccc} \mathcal{A} & \simeq & \mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et]) \\ \downarrow & & \downarrow \\ [\mathcal{I}nj_0, \mathcal{S}et] & \xrightarrow[J_*]{} & [\mathcal{C}tx_0, \mathcal{S}et] \end{array} \tag{6}$$

Proof notes: We begin by characterizing the right adjoint $J_* : [\mathcal{I}nj_0, \mathcal{S}et] \to [\mathcal{C}tx_0, \mathcal{S}et]$. By the Yoneda lemma,

$$\begin{aligned} (J_*X)(p) &\cong [\mathcal{I}nj_0, \mathcal{S}et](J^*(\mathcal{C}tx_0(p, -)), X) \\ &\cong [\mathcal{I}nj_0, \mathcal{S}et](\textstyle\sum_m S_p(m) \times \mathcal{I}nj_0(m, -), X) \\ &\cong \textstyle\prod_m X(m)^{S_p(m)} \end{aligned} \tag{7}$$

using the Stirling numbers $S_p(m)$. The bijection $\mathcal{C}tx_0(p, n) \cong \sum_m S_p(m) \times \mathcal{I}nj_0(m, n)$ is standard (e.g. [54, §1.9],[17]).

It follows from (7) that J_* preserves filtered colimits and reflexive coequalizers, since they commute with products in $\mathcal{S}et$. By direct calculation, J_* reflects isomorphisms between 1-inhabited presheaves. We use this to deduce that the composite $\mathcal{A} \to [\mathcal{I}nj_0, \mathcal{S}et] \to [\mathcal{C}tx_0, \mathcal{S}et]$ is $[\mathcal{C}tx_0, \mathcal{S}et]$-enriched finitary crude-monadic. ∎

To some extent we can go the other way:

Proposition 20. *If a parameterized algebraic theory $\mathbb{T}$ contains the theory of restriction with equality testing then there is a $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched finitary crude-monadic functor $U : \mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et]) \to [\mathcal{I}nj_0, \mathcal{S}et]$ making the following diagram commute:*

$$\begin{array}{ccc} & & \mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et]) \\ & \swarrow & \searrow \\ [\mathcal{I}nj_0, \mathcal{S}et] & \xrightarrow[J_*]{} & [\mathcal{C}tx_0, \mathcal{S}et] \end{array} \tag{8}$$

Proof notes: Since $J^* : [\mathcal{I}nj_0, \mathcal{S}et] \to [\mathcal{C}tx_0, \mathcal{S}et]$ preserves products, we can consider $\mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et])$ as a $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched category. The composite functor $\mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et]) \to \mathcal{M}od(\nu,?_=; [\mathcal{C}tx_0, \mathcal{S}et]) \to [\mathcal{I}nj_0, \mathcal{S}et]$ is $[\mathcal{I}nj_0, \mathcal{S}et]$-enriched finitary crude-monadic. ∎

Note: Our results in this section are for models of parameterized algebraic theories in $[\mathcal{C}tx_0, \mathcal{S}et]$. It is typically not the case that $\mathcal{M}od(\mathbb{T}; [\mathcal{C}tx_0, \mathcal{S}et]) \simeq \mathcal{M}od(\mathbb{T}; [\mathcal{I}nj_0, \mathcal{S}et])$.

3) Aside: Each of the three monads on $[\mathcal{I}nj_0, \mathcal{S}et]$ is strong and preserves filtered colimits and reflexive coequalizers. Thus, by Proposition 17, they correspond to abstract clones in $\mathcal{I}nj_1 \subseteq [\mathcal{I}nj_0, \mathcal{S}et]$, where $\mathcal{I}nj_1$ is the finite coproduct completion of $\mathcal{I}nj_0$. The category $\mathcal{I}nj_1$ is widely regarded as important. It can be described in terms of 'strong nominal sets' [60] or 'named sets' [46], and it has already been used as arities: for graded Lawvere theories (Melliès, [38]), for the π-calculus (Lack and Rosický, [32, Ex. 5.16]), and for a completeness result for local store [58, Thm. 6].

However, there is still no compelling general syntactic framework for abstract clones in $\mathcal{I}nj_1 \subseteq [\mathcal{I}nj_0, \mathcal{S}et]$. (This is what led the author to parameterized algebraic theories.) We note in passing that there is a non-cartesian monoidal closed structure on $[\mathcal{I}nj_0, \mathcal{S}et]$ which is regarded as important in modelling variable binding ([18],[25]) and separation in local store (e.g. [44],[43],[51]). This closed structure is the focus of Melliès' string diagrams [38] and plays an important role in nominal equational logic ([12], [18], [21], [31]).

Acknowledgment

Thanks to an anonymous reviewer for suggestions and to Danel Ahman, Marcelo Fiore, Paul Levy, Paul-André Melliès, Rasmus Møgelberg, Gordon Plotkin, John Power, Alex Simpson and Ian Stark for helpful discussions.

References

[1] J. Aczél, "On mean values," *Bull. AMS*, vol. 54, pp. 392–400, 1948.
[2] P. Aczel, "A general Church-Rosser theorem," 1978.
[3] J. Adamek, J. Rosicky, and E. M. Vitale, "What are sifted colimits?" *Theory Appl. Categ.*, vol. 23, no. 13, pp. 251–260, 2010.
[4] J. Adámek and J. Rosický, "On sifted colimits and generalized varieties," *Theory Appl. Categ.*, vol. 8, no. 3, pp. 33–53, 2001.
[5] R. Adams, "Lambda-free logical frameworks," *Ann. Pure Appl. Logic*, to appear.
[6] T. Altenkirch, J. Chapman, and T. Uustalu, "Monads need not be endofunctors," in *FOSSACS'10*, 2010, pp. 297–311.
[7] M. Barr and C. Wells, *Toposes, triples and theories*. Springer, 1985.
[8] C. Berger, P.-A. Melliès, and M. Weber, "Monads with arities and their associated theories," *J. Pure Appl. Algebra*, vol. 216, 2012.
[9] R. S. Bird, "Lectures on constructive functional programming," 1988.
[10] A. Carboni and P. Johnstone, "Connected limits, familial representability and Artin glueing," *Math. Struct. Comput. Sci.*, vol. 5, no. 4, 1995.
[11] R. Clouston, "Nominal lawvere theories," in *WoLLIC 2011*, pp. 67–83.
[12] R. A. Clouston and A. M. Pitts, "Nominal equational logic," in *Computation, Meaning, and Logic*. Elsevier, 2007.
[13] P. M. Cohn, *Universal algebra*, 2nd ed. D Reidel, 1981.
[14] P.-L. Curien, "Operads, clones and distributive laws," in *Operads and Universal Algebra*. World Scientific, 2012.
[15] T. Evans, "Abstract mean values," *Duke Math. J.*, vol. 30, 1963.
[16] M. P. Fiore and C.-K. Hur, "Second-order equational logic," in *Proc. CSL'10*, 2010.
[17] M. Fiore, "Notes on combinatorial functors (draft)," 2001.
[18] M. P. Fiore and C.-K. Hur, "Term equational systems and logics," in *Proc. MFPS 2008*, 2008, pp. 171–192.
[19] M. P. Fiore and D. Turi, "Semantics of name and value passing," in *Proc. LICS'01*, 2001, pp. 93–104.
[20] M. J. Gabbay, T. Litak, and D. Petrisan, "Stone duality for nominal boolean algebras with new," in *CALCO'11*, 2011, pp. 192–207.
[21] M. J. Gabbay and A. Mathijssen, "Nominal (universal) algebra: Equational logic with names and binding," *J. Log. Comput.*, vol. 19, 2009.
[22] J. Gibbons and R. Hinze, "Just do it: simple monadic equational reasoning," in *ICFP*, 2011, pp. 2–14.
[23] M. Hamana and M. Fiore, "Multiversal polymorphic algebraic theories," in *To appear in Proc. LICS 2013*, 2013.
[24] R. Heckmann, "Probabilistic domains," in *CAAP 1994*, pp. 142–156.
[25] M. Hofmann, "Semantical analysis of higher-order abstract syntax," in *Proc. LICS'99*, 1999, pp. 204–213.
[26] M. Hyland, "Multicategories in and around algebra and logic," invited talk, TACL'09.
[27] O. Kammar and G. D. Plotkin, "Algebraic foundations for effect-dependent optimisations," in *Proc. POPL'12*, 2012.
[28] G. M. Kelly, *Basic Concepts of Enriched Category Theory*. CUP, 1982.
[29] ——, "Structures defined by limits in the enriched context," *Cah. Topologie Géom. Différ. Catégoriques*, vol. 23, pp. 3–42, 1982.
[30] G. M. Kelly and A. J. Power, "Adjunctions whose counits are coequalisers," *J. Pure Appl. Algebra*, vol. 89, pp. 163–179, 1993.
[31] A. Kurz and D. Petrişan, "Presenting functors on many-sorted varieties and applications," *Inform. Comput.*, vol. 208, pp. 1421–1446, 2010.
[32] S. Lack and J. Rosický, "Notions of Lawvere theory," *Appl. Categ. Structures*, vol. 19, no. 1, 2011.
[33] Y. Lafont, B. Reus, and T. Streicher, "Continuations semantics or expressing implication by negation," 1993, Munich.
[34] P. B. Levy, J. Power, and H. Thielecke, "Modelling environments in call-by-value programming languages," *Inform. Comput.*, 2003.
[35] F. E. J. Linton, "Autonomous equational categories," *J. Math. Mech.*, vol. 15, pp. 637–642, 1966.
[36] S. Lösch and A. M. Pitts, "Relating two semantics of locally scoped names," in *Proc. CSL'11*, 2011.
[37] P.-A. Melliès, "Segal condition meets computational effects," in *LICS 2010*.
[38] ——, "Local stores in string diagrams," `tinyurl.com/mellies-itu-2011`.
[39] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes, I&II," *Inf. Comput.*, vol. 100, no. 1, pp. 1–77, 1992.
[40] R. E. Møgelberg and S. Staton, "Linearly-used state in models of call-by-value," in *CALCO 2011*.
[41] E. Moggi, "An abstract view of programming languages," 1989.
[42] ——, "Notions of computation and monads," *Inform. Comput.*, vol. 93, no. 1, 1991.
[43] P. W. O'Hearn and R. D. Tennent, Eds., *Algol-like Languages*, 1997.
[44] P. O'Hearn, "On bunched typing," *J. Funct. Programming*, vol. 13, no. 4, pp. 747–796, 2003.
[45] J. Parrow and D. Sangiorgi, "Algebraic theories for name-passing calculi," *Inf. Comput.*, vol. 120, no. 2, pp. 174–197, 1995.
[46] M. Pistore, "History dependent automata," Ph.D. dissertation, Univ. Pisa, 1999.
[47] A. M. Pitts, "Structural recursion with locally scoped names," *JFP*, 2011.
[48] G. Plotkin, "Some varieties of equational logic." Springer, 2006.
[49] G. D. Plotkin and J. Power, "Notions of computation determine monads," in *Proc. FOSSACS'02*.
[50] ——, "Algebraic operations and generic effects," *Appl. Categ. Structures*, vol. 11, no. 1, pp. 69–94, 2003.
[51] J. Power, "Indexed Lawvere theories for local state," in *Models, Logics and Higher- Dimensional Categories*. AMS, 2011, pp. 268–282.
[52] ——, "Semantics for local computational effects," in *Proc. MFPS'06*.
[53] ——, "Enriched Lawvere theories," *Theory Appl. Categ.*, vol. 6, 1999.
[54] R. P. Stanley, *Enumerative Combinatorics*. CUP, 2011, vol. 1.
[55] I. Stark, "Free-algebra models for the π-calculus," in *FOSSACS'05*.
[56] ——, "Categorical models for local names," *LISP and Symb. Comp.*, vol. 9, no. 1, pp. 77–107, 1996.
[57] ——, "Free-algebra models for the pi-calculus," *Theor. Comput. Sci.*, vol. 390, no. 2-3, pp. 248–270, 2008.
[58] S. Staton, "Completeness for algebraic theories of local state," in *Proc. FOSSACS'10*.
[59] ——, "An algebraic presentation of predicate logic," in *FOSSACS 2013*, 2013.
[60] N. Tzevelekos, "Full abstraction for nominal general references," in *Proc. LICS'07*, 2007, pp. 399–410.
[61] T. Uustalu, "Strong relative monads," in *Proc. CMCS 2010*.