A Relational Theory of Monadic Rewriting Systems, Part i

A Relational Theory of Monadic Rewriting Systems, Part I

Claudia Faggian
Laboratoire IRIF - Équipe PPS
University of Paris

Francesco Gavazzo
University of Bologna
INRIA Sophia Antipolis

***Abstract*—Motivated by the study of effectful programming languages and computations, we introduce a relational theory of monadic rewriting systems. The latter are rewriting systems whose notion of reduction is effectful, where effects are modelled as monads. Contrary to what happens in the ordinary operational semantics of monadic programming languages, defining meaningful notions of monadic rewriting turns out to problematic for several monads, including the distribution, powerset, reader, and global state monad. This raises the question of *when* monadic rewriting is possible. We answer that question by identifying a class of monads, known as weakly cartesian monads, that guarantee monadic rewriting to be well-behaved. In case monads are given as equational theories, as it is the case for algebraic effects, we also show that a sufficient condition to have well-behaved notion of monadic rewriting is that all equations in the theory are linear. Finally, we apply the abstract theory of monadic rewriting systems to the call-by-value λ-calculus with algebraic effects, this way obtaining effectful (surface) standardisation and confluence theorems.**

I. Introduction

Rewriting Theory [1] is the discipline that studies the general properties of object transformations [2]. Among its many applications, rewriting theory plays a central role in computer science, where it is used both as a foundational theory of computation and as a formalism for the development of programming languages. To have a taste of the role played by rewriting theory, we consider the λ-calculus [3], which is arguably the most well-known rewriting system in theoretical computer science. The λ-calculus provides both a model of computation and a foundation of *pure* higher-order sequential programming languages, with fundamental results, such as the Church-Rosser theorem, ensuring the soundness of powerful optimisation techniques, such as parallelisation. These results play a crucial role in the development of compilers, where efficiency can be gained by changing the order of standard evaluation strategies [4], or even by leaving such strategies unspecified until runtime, when they are chosen according to the available resources [5]. None of these techniques would be sound without results such as the Church-Rosser theorem.

Nowadays programming languages, however, *are not pure*. Quite the opposite, actually: they are deeply *effectful*, with programs exhibiting several computational effects at once (raising exceptions, performing input-output operations, sampling from distributions, etc). This makes the λ-calculus and, more generally, the ordinary theory of rewriting systems inadequate to understand such languages and their associated notion of computation.

Starting with the seminal work by Moggi [6], [7], monads [8] have been proved to be the right mathematical abstraction for the denotational analysis of computational effects, whereas Plotkin and Power [9]–[11] showed that an *operational* analysis of effects can be given in terms of equational theories, this way introducing the theory of *algebraic effects*. Since their introduction, both monads and algebraic effects have been extensively investigated in terms of programming languages and their semantics: but what about rewriting systems?

To the best of the authors' knowledge, only specific computational effects have been studied in the setting of rewriting theory (the main example being the recent theory of probabilistic rewriting [12]–[16]) and nothing has been done at the general level of monadic or algebraic effects. This is rather unsatisfactory, as a proper analysis of effectful programming languages and computations requires a general theory of *effectful rewriting*. For instance, we may want to know whether the optimisation techniques enabled by the Church-Rosser theorem in the pure λ-calculus continue to hold in presence of probabilistic nondeterminism, or in presence of global stores, or both. More generally, we would like to isolate classes of effects for which a given (optimisation) technique is sound.

This paper aims to answer those questions by making a first step towards the development of a general, *relational theory of monadic rewriting systems*. The development of such a theory turns out to be nontrivial since, contrary to what happens in the ordinary operational semantics of programming languages with monadic and algebraic effects [10], we discovered that (relational) rewriting is simply not possible for several monads, including the distribution, powerset, reader, and global state monad. This raises the question of *when* monadic rewriting is possible. We answer that question by identifying a class of monads, known as *weakly cartesian monads* [17]–[20], that support (monadic) rewriting. Moreover, in case effects are defined by means of equational theories (as it is for algebraic effects), we give a sufficient condition on the shape of a theory that makes its associated monadic rewriting well-behaved: such a condition is based on an old theorem by Gautam [21] about power algebras [22], and requires all equations of the theory to be *linear*. This result also sheds some light upon what researchers have been done in specific cases of monadic rewriting (viz. probabilistic rewriting). Finally, we witness

the effectiveness of our analysis by instantiating our theory of monadic rewriting to Plotkin's call-by-value λ-calculus [23] extended with algebraic effects. As a byproduct of that, we obtain two new theorems: (surface) standardisation and confluence. Remarkably, the former holds for *all* algebraic effects (admitting well-behaved monadic rewriting), whereas the latter requires effects to be commutative, a well-known notion in programming language theory.

Summing up, the contributions of this work are:

1) A novel notion of a *monadic rewriting system* based on the notion of a relational extension [24]–[26].
2) The identification of a class of monads — namely the class of *weakly cartesian monads* [17]–[20] — for which monadic rewriting is indeed possible.
3) The development of nontrivial proof principles for reasoning about monadic rewriting systems.
4) An analysis of monadic rewriting in terms of equational theories and a sufficient condition on the shape of a theory that makes monadic rewriting well-behaved.
5) The application of the general theory of monadic rewriting systems to the call-by-value λ-calculus with algebraic effects, and the proof of effectful (surface) standardisation and confluence for such a calculus.

A. A Roadmap

We expand on the main contributions of this work.

a) Monadic Rewriting Systems: The first goal of this work is to define the notion of a *monadic (abstract) rewriting system* (MARS), i.e. of an effectful rewriting system whose computational effects are given in terms of monads. Our definition is *relational*, whereby a MARS is given by a set of objects together with a rewriting relation between objects and *monadic* elements. That, however, is not enough: in order to study rewriting sequences, we need a way to compose monadic rewriting steps, which essentially amounts to give a *relational extension* [24]–[27] of the underlying monad, an important concept studied in fields such as topology [24], [26], [27], coalgebra [28]–[31], and programming language semantics [32]–[37]. MARSs are introduced in Section IV.

b) Negative Results: A MARS is thus parametric with respect to a monad (which specifies the kind of effects produced) and a relational extension (which specifies how rewriting steps are composed). To qualify those relational extensions that give meaningful notions of rewriting, we introduce the (minimal) properties of *compatibility* and *preservation*, and argue that both these properties are necessary to define meaningful MARSs. Perhaps surprisingly, we discover that such properties are incompatible with several monads — including the distribution, reader, powerset, and global state monads — that are regularly used to model computational effects: for such monads, there is no relational extension satisfying both compatibility and preservation. Compatibility, preservation, and negative results are given in Section II, where we give a gentle introduction to MARSs by means of a simple example.

c) Weakly Cartesian Monads and Relational Extensions: In light of the aforementioned negative result(s), the most urgent question to be answered becomes: *what are the monads for which monadic rewriting is possible?* We answer this question by showing that weakly cartesian monads are precisely those monads on which one can do monadic rewriting. Moreover, a classic result by Barr [24] states that not only for such monads there exists a well-behaved relational extension, but such an extension is also unique and it is given by the so-called Barr extension of the monad [24]–[26]. Additionally, we show that the axiomatics of a weakly cartesian monad entails the soundness of powerful rewriting techniques that simplify the analysis of MARSs. As a main result in this respect, Theorem 22 shows the diamond property is preserved by Barr extensions. Weakly cartesian monads and the Barr theorem are the subject of Section IV-A, whereas monadic rewriting techniques are studied in Section IV-B.

d) Algebraic Theories: When dealing with algebraic effects, monads are given as equational or algebraic theories. In those cases, it is natural to ask whether it is possible to find a sufficient condition on the shape of an equational theory that ensures the corresponding monadic rewriting to be possible. We answer that question in the affirmative, by showing that for all *linear* equational theories monadic rewriting is indeed possible. This follows from a classic theorem by Gautam [21]. Monadic rewriting on algebraic theories is studied in Section IV-C.

e) The λ-calculus with Algebraic Operations: Finally, we use our general theory of MARSs to make an operational analysis of the call-by-value λ-calculus with algebraic effects. We prove two new theorems: effectful *surface standardisation* (Theorem 38) and *confluence* (Theorem 43). The former holds for *all* algebraic effects (making monadic rewriting well-behaved), whereas the latter requires effects to be *commutative*. This is done in Section V.

II. Monadic Rewriting, An Invitation

In this section, we give an informal introduction to the relational theory of monadic abstract rewriting (or reduction) systems (MARSs) by means of a concrete example: probabilistic rewriting systems [12], [13], [15].

Recall that an abstract rewriting (or reduction) system (ARS) is a structure (A, R) consisting of a set of objects A together with a rewrite relation $R : A \nrightarrow A$.[1] A probabilistic rewriting system is an extension of an ARS where the rewriting relation has a probabilistic behaviour. Probabilistic nondeterminism is usually modelled using the *distribution monad*, but for our purposes the *finite distribution monad* is sufficient. Given a set X, we denote by $\mathcal{D}X$ the collection of finite distribution on X, i.e. the set of functions $\varphi : X \to [0, 1]$ such that: *(i)* $\varphi(x) \neq 0$ for finitely many x only; *(ii)* $\sum_x \varphi(x) = 1$. We oftentimes denote a distribution φ as a formal sum $\sum_{i \in I} p_i{:}x_i$, remarking that such a notation is *not* unique. The construction $\mathcal{D}$ gives a functor which we endow with a monad structure $\mathbb{D} = (\mathcal{D}, \eta^{\mathbb{D}}, \mu^{\mathbb{D}})$. The unit $\eta^{\mathbb{D}}(x)$ maps each $y \in X$

[1] We write $R : X \nrightarrow Y$ for relations R over $X \times Y$ and $R; S : X \nrightarrow Z$ for the composition of two relations $R : X \nrightarrow Y$, $S : Y \nrightarrow Z$.

to 0 if $x \neq y$, and to 1 otherwise; the multiplication maps a distribution $\Phi \in \mathcal{D}\mathcal{D}X$ to the distribution $\mu^{\mathbb{D}}(\Phi)$ defined by $\mu^{\mathbb{D}}(\Phi)(x) = \sum_{\varphi \in \mathcal{D}X} \Phi(\varphi)\varphi(x)$. As it is customary when representing distributions as formal sums, we write $\sum_i p_i{:}\varphi_i$ in place of $\mu^{\mathbb{D}}(\sum_i p_i{:}\varphi_i)$.

Following Bournez and Garnier [13], a probabilistic rewriting system is thus defined as a set A together with a *probabilistic* rewrite relation $R : A \nrightarrow \mathcal{D}A$. Given such a system, we are interested to study how the system evolves, i.e. we want to study its rewriting sequences. At this point, the literature offers two possibilities.

1) The first one [13] studies probabilistic rewriting sequences by means of Markov decision processes, and thus focuses on the stochastic evolution of *single* rewriting sequence.
2) The second one [14]–[16] analyses the *global* behaviour of probabilistic systems by lifting the underlying rewriting relation $R : A \nrightarrow \mathcal{D}A$ to a relation acting on a *variation* of distributions, viz. multi-distributions, this way obtaining an ARS whose objects are themselves multi-distributions.

Although those approaches are equivalent [15], we focus on the second one, the latter being truly *relational*. That allows us to stay closer to the definition of an ARS, and thus to extend theories and techniques developed in a pure setting to a probabilistic scenario.

But why does one need to work with a *variation* of distributions? Naively, one would expect to lift the rewriting relation $R : A \nrightarrow \mathcal{D}A$ to a relation $\Gamma R : \mathcal{D}A \nrightarrow \mathcal{D}A$ on distributions, and to study the ARS $(\mathcal{D}A, \Gamma R)$. Avanzini, Dal Lago, and Yamada [15], however, showed that such an approach does not work well: working with distributions, one naturally looses some rewriting sequences, this way creating a discrepancy with the first approach in the aforementioned list. Avanzini et al. overcame the problem by moving from distributions to *multi-distributions* (see Section III), this way obtaining the desired equivalence. Here, we attack the problem from a more foundational perspective: first, we propose two minimal conditions on (candidate) relation liftings and argue that such conditions are necessary to have a meaningful notion of a probabilistic rewriting system; secondly, we show that there exists *no* relation lifting satisfying such conditions. This negative result is the deep reason why Avanzini et al. define probabilistic rewriting systems relying on multi-distributions. Our two conditions are:

1) The relation ΓR must be *compatible* with the (probabilistic) effects produced during the rewriting process;
2) Γ must *preserve* pure rewriting back and forth.

Let us expand on that.

a) Compatibility: Compatibility means that the relation ΓR can always apply R inside convex combinations or, more formally, that ΓR is closed under convex combinations. This is a natural, minimal condition ensuring that, e.g., no matter how Γ is defined, if $1{:}a$ rewrites into $\frac{1}{3}{:}c + \frac{2}{3}{:}c'$ and $1{:}b$ rewrites into $1{:}c$, then $\frac{1}{2}{:}a + \frac{1}{2}{:}b$ rewrites into $\frac{1}{6}{:}c + \frac{1}{3}{:}c' + \frac{1}{2}{:}d$. Formally, compatibility states that if $\varphi_1 \; \Gamma R \; \psi_1, \ldots, \varphi_n \; \Gamma R \; \psi_n$, then $\sum_i p_i{:}\varphi_i \; \Gamma R \; \sum_i p_i{:}\psi_i$, for any convex combination $\sum_i p_i$.

b) Preservation: Preservation is a more delicate property than compatibility, and it often makes effectful rewriting problematic. Preservation ensures that programs with *no* probabilistic operation do *not* produce nontrivial probabilistic behaviours. More precisely, given a probabilistic rewriting system $(A, R : A \nrightarrow \mathcal{D}A)$, preservation states that performing a R-rewriting step on an element a is the same as performing a ΓR-rewriting step on $1{:}a$. Formally, $a \; R \; \varphi$ iff $(1{:}a) \; \Gamma R \; \varphi$. This property ensures that, e.g., any *non-probabilistic* λ-term $\mathtt{t}$ can be studied in a probabilistic setting as $1{:}\mathtt{t}$. For instance, preservation ensures that the pure λ-term $\mathtt{t} = (\mathtt{Ix})(\mathtt{Iy})$, where $\mathtt{I} = \lambda\mathtt{x}.\mathtt{x}$, has *only* the following *trivially* probabilistic reductions (we write $\Rightarrow_\beta$ in place of $\Gamma(\rightarrow_\beta)$ [16]):

$$1{:}\mathtt{t} \Rightarrow_\beta 1{:}\mathtt{x}(\mathtt{Iy}) \Rightarrow_\beta 1{:}\mathtt{xy} \qquad 1{:}\mathtt{t} \Rightarrow_\beta 1{:}(\mathtt{Ix})\mathtt{y} \Rightarrow_\beta 1{:}\mathtt{xy}.$$

In particular, preservation entails that the operational behaviour of $\mathtt{t}$ in a *pure* setting is essentially the same as the one of $1{:}\mathtt{t}$ in a probabilistic setting, meaning that pure programs cannot produce nontrivial probabilistic behaviours. Please notice that without preservation, nothing will prevent (in principle) $1{:}\mathtt{t}$ to rewrite into, e.g., $\frac{3}{8}{:}\mathtt{x}(\mathtt{Iy}) + \frac{5}{8}{:}(\mathtt{Ix})\mathtt{y}$. This behaviour may seem *prima facie* innocent, as a further rewriting step gives the distribution $1{:}\mathtt{xy}$. However, what would happen if instead of having the redexes $\mathtt{Ix}$, $\mathtt{Iy}$ one has two terms performing two different output actions? In that case, we would have that a non-probabilistic program generates a truly probabilistic behaviour which, thanks to the presence of output, is now visible to an external observer. This is definitely something undesired.

c) No Extension for the Distribution Monad: We have identified compatibility and preservation as two minimal conditions any probabilistic rewriting system should satisfy to be considered operationally meaningful. As a further evidence of that, we show that compatibility and preservation are necessary conditions to make probabilistic rewriting *mathematically* well-defined. To understand that, let us recall that in an ARS (A, R), a reduction sequence can be ultimately described as a sequence $R; R; \cdots; R$ of compositions of the rewriting relation R with itself. If the system is probabilistic, this definition is not possible anymore. However, the lifting Γ naturally induces a notion of composition on relations of type $A \nrightarrow \mathcal{D}A$ as $R \cdot S = R; \Gamma S$. A (probabilistic) rewriting sequence can be thus defined as a sequence of the form $R \cdot R \cdot \cdots \cdot R$. Following the analogy with ARSs, we would like the composition operation $\cdot$ to behave as be as close as possible to ordinary relation composition. In particular, $\cdot$ must be associative and have a neutral element, which we expect to be the unit $\eta : A \rightarrow \mathcal{D}A$ of the distribution monad (regarded as a relation).[2] It turns out that these desiderata cannot hold if preservation and compatibility are not satisfied.

[2]Cf. with the notion of a Kleisli composition [6], [8].

Proposition 1. *If compatibility or preservation does not hold, then one of the following laws does not hold too:*

$$R \cdot \eta = R \qquad \eta \cdot S = S \qquad R \cdot (S \cdot P) = (R \cdot S) \cdot P.$$

That is, without compatibility and preservation, there is no hope to obtain a meaningful notion of rewriting sequence relying on Γ. Proposition 1 is a direct corollary of Proposition 15, which establishes a tighter connection between relation lifting and composition of (monadic) rewriting relations.

Given the importance of compatibility and preservation, a natural question arises: *can we find a lifting Γ satisfying both compatibility and preservation?* The answer is in the negative.

Proposition 2. *There is no lifting Γ satisfying both compatibility and preservation*

Proof. Suppose to have such a lifting Γ, and consider the probabilistic rewriting system defined by the relation $a\,R\,1{:}b$, $a\,R\,1{:}c$, with $a \neq b \neq c$. Then, preservation implies $1{:}a\,\Gamma R\,1{:}b$ and $1{:}a\,\Gamma R\,1{:}c$, from which follows, by compatibility: $1{:}a = (\frac{1}{2}{:}a + \frac{1}{2}{:}a)\,\Gamma R\,(\frac{1}{2}{:}b + \frac{1}{2}{:}c)$. Again, by preservation we should have $a\,R\,(\frac{1}{2}{:}b + \frac{1}{2}{:}c)$ which, however, is not the case. Hence such an extension Γ does not exist. □

Proposition 2 gives a strong, negative result: probabilistic rewriting as we have defined it, is simply not possible. Moreover, this does not concern the lifting Γ we choose, but it is an intrinsic trait of the distribution monad.

d) The Anatomy of the Problem: Where do problems with the distribution monad come from? The proof of Proposition 2 clearly shows that the crucial passage is moving from $1{:}a$ to $\frac{1}{2}{:}a + \frac{1}{2}{:}a$, which is an instance of the *idempotency law* characterising the algebra of formal sums. Such a law gives $\mathcal{D}$ a form of nondeterminism that does not interact well with the implicit nondeterminism of the very act of rewriting. From a more abstract perspective, any probabilistic (rewriting) relation $R : X \nrightarrow \mathcal{D}X$ can be seen as a function $r : X \to \mathcal{P}\mathcal{D}X$. Since there is a one-to-one correspondence between well-behaved liftings and distributive laws [38] of type $\delta : \mathcal{D}\mathcal{P} \Rightarrow \mathcal{P}\mathcal{D}$ (see Remark 20), we can read Proposition 2 as an instance of the well-known result that there is no distributive law of the distribution monad over the powerset monad [39], [40].

But does that mean that a relational theory of probabilistic rewriting is not possible at all? Not really, it means that (full) probabilistic rewriting cannot be done relying on the distribution monad, although we use the latter to model probabilistic computations. This is the deep reason why relational approaches to probabilistic rewriting [14]–[16] rely on the *multi-distribution* monad.[3] And in fact, the multi-distribution monad is *de facto* obtained from the distribution monad by removing the idempotency equation from the algebra of formal sums.

[3]Actually, the very same monad has been introduced before [39], [40] under the name of *indexed valuations monad*, where it is also proved the existence of a distributive law of the indexed valuation monad over the powerset monad.

e) No-go Theorems for Monadic Rewriting: Summing up, we have identified two minimal *necessary* criteria to have a meaningful (relational) notion of probabilistic rewriting and have showed that as long as we use the distribution monad, there is no hope to have such a notion. The next step is to realise that our negative results do not affect the distribution monad only. In fact, Proposition 2 can be easily modified to the case of the powerset monad $\mathcal{P}$, the source of the problem being again the idempotency law $x \cup x = x$; and indeed, the powerset monad does not distribute over itself [41]. Furthermore, even the group monad suffers the same pathology (even without idempotency, we can rely on the equations $0 = x + (-x)$ and $x = x + 0$), as well as the reader and global state monads.

f) Monadic Rewriting Systems: The negative results seen so far show that developing a well-behaved relational theory of monadic rewriting systems is problematic: for several monads that are used to model computational effects, monadic rewriting turned out to be simply not possible. Of course, here we mean monadic rewriting as constrained by compatibility and preservation; but we argued that rewriting systems not satisfying such constraints can be hardly judged to be operationally meaningful (as well as mathematically well-behaved, as they might give, e.g., non-associative rewriting sequences). These observations show that a proper understanding of monadic computations also goes through the development of a theory of monadic rewriting systems, the task of such a theory being to precisely define what a monadic rewriting system is, to isolate the class of monads for which monadic rewriting is indeed possible, and to develop a collection of techniques for studying such systems. That is our plan for this work.

III. Preliminaries: Monads, Algebra, and Rewriting

In this section, we recall some preliminary notions on monads [8], algebras [42], and rewriting [1] that will be central in the rest of this paper. Due to space constraints, there is no hope to be comprehensive and thus we assume the reader to have a minimal familiarity with those fields. Unless explicitly stated, we work in the category $\mathbf{Set}$ of sets and functions, and we tacitly restrict all definitions to it.

A. Monads and Algebraic Effects

Starting with the seminal work by Moggi [6], [7], *monads* have become a standard formalism to model and study computational effects. We recall some basic notions about monads and algebraic operations.

Definition 3. *A* monad *(on $\mathbf{Set}$) is a triple*[4] $\mathbb{T} = (T, \eta^{\mathbb{T}}, \mu^{\mathbb{T}})$ *consisting of a functor T (on $\mathbf{Set}$) together with two natural transformations: $\eta^{\mathbb{T}} : 1_{\mathbf{Set}} \Rightarrow T$ (called* unit*) and $\mu^{\mathbb{T}} : TT \Rightarrow T$ (called* multiplication*) subject to the following laws: $\mu^{\mathbb{T}} \circ \eta^{\mathbb{T}}T = \mu^{\mathbb{T}} \circ T\eta^{\mathbb{T}} = 1_T$ and $\mu^{\mathbb{T}} \circ T\mu^{\mathbb{T}} = \mu^{\mathbb{T}} \circ \mu^{\mathbb{T}}T$.*

[4]When unambiguous, we omit superscripts from η and μ.

Any monad $\mathbb{T}$ defines two useful categories: the category of Eilenberg-Moore algebras of $\mathbb{T}$ and the Kleisli category of $\mathbb{T}$. We denote them by $\mathbf{EM}(\mathbb{T})$ and $\mathbf{Kl}(\mathbb{T})$, respectively.[5]

Monads model notions of computational effects. To model how actual effects are produced, Plotkin and Power [9], [10] introduced the notion of an *algebraic operation*.

Definition 4. *Given a monad $\mathbb{T} = (T, \eta, \mu)$, an n-ary* algebraic operation *is a natural transformation $f : T^n \Rightarrow T$ respecting the monad multiplication.*

From an operational perspective, algebraic operations describes those operations whose execution is independent of the context in which they are executed.

Example 5. Throughout this paper, we will use the following monads as running examples.

1) Divergent computations are modelled by the *maybe* or *partiality* monad $\mathbb{M} = (\mathcal{M}, \eta^{\mathbb{M}}, \mu^{\mathbb{M}})$, where $\mathcal{M}X = X + \{\bot\}$, $\eta^{\mathbb{M}}$ is the left injection ι_ℓ, and $\mu^{\mathbb{M}} : ((X + \{\bot\}) + \{\bot\}) \to X + \{\bot\}$ sends $\iota_\ell(\iota_\ell(x))$ to $\iota_\ell(x)$, and all the rest to $\bot$.
2) Probabilistic computations are modelled by the (finitary) distribution monad as seen in Section II.
3) Computations with output are modelled by the *writer* (sometimes called *output* or *action*) monad $\mathbb{W} = (\mathcal{W}, \eta^{\mathbb{W}}, \mu^{\mathbb{W}})$, where $\mathcal{W}X = W \times X$ and $(W, 1, \cdot)$ is a monoid. Unit and multiplication are defined by $\eta^{\mathbb{W}}(x) = (1, x)$ and $\mu^{\mathbb{W}}(w, (v, x)) = (w \cdot v, x)$. Taking the monoid of words, then we read (w, x) as the result of a program printing w and returning x. If, instead, we take the monoid $(\mathbb{N}, 0, +)$, then we obtain the *cost* or *complexity* monad, whereby we read (n, x) as the result of a computation that produces x with cost n. We consider a W-indexed family of unary operations $\mathtt{out}_w$ defined by $\mathtt{out}_w(v, x) = (w \cdot v, x)$. These are indeed algebraic.
4) We model computations reading from a set of states using the *reader* monad. For simplicity, we consider a single location containing values from a (finite) set R only. In this case, the reader monad is the triple $\mathbb{R} = (\mathcal{R}, \eta^{\mathbb{R}}, \mu^{\mathbb{R}})$, where $\mathcal{R}X = X^R$, $\eta^{\mathbb{R}}(x)(r) = x$, and $\mu^{\mathbb{R}}(f)(r) = (f(r))(r)$. An element f in $\mathcal{R}X$ represent a computation that behaves as $f(r)$ if the location contains r.
5) We model nondeterminism using the powerset monad $\mathbb{P} = (\mathcal{P}, \eta^{\mathbb{P}}, \mu^{\mathbb{P}})$, where $\eta^{\mathbb{P}}(x) = \{x\}$ and $\mu^{\mathbb{P}}(\mathcal{U}) = \bigcup \mathcal{U}$. We generate nondeterminism using (binary) set-theoretic union, which is indeed algebraic. The finitary powerset monad $\mathbb{P}_f$ is obtained from $\mathbb{P}$ by taking as underling functor the finite powerset functor $\mathcal{P}_f$. Similarly, the non-empty powerset monad $\mathbb{P}^+$ is obtained by the taking the non-empty powerset functor $\mathcal{P}^+$.

[5]We will use these notions only in few technical passages in Section IV. The reader not familiar with these concepts will still be able to follow the paper. Here we mentioned them only to fix the notation.

B. Algebraic Theories

Since effects are ultimately produced by algebraic operations, we oftentimes describe computational effects by means of *algebraic theories*, i.e. via a collection of operations and equations.

Definition 6. *Recall that a* signature Σ *is a family of sets $\{\Sigma_k\}_{k \in \mathbb{N}}$, the elements $\sigma, \rho, \ldots$ of each Σ_k being called k-ary* operations.

1) *The set $T_\Sigma X$ of* Σ-terms *(just terms if unambiguous) over X (notation $t, s, \ldots$) is inductively defined thus: $x \in T_\Sigma X$, for any $x \in X$; $\sigma(t_1, \ldots, t_n) \in T_\Sigma X$, whenever $t_1, \ldots, t_n \in T_\Sigma X$.*
2) *A Σ-algebra $\mathbb{A} = (A, \{\sigma_{\mathbb{A}}\}_{\sigma \in \Sigma})$ is given by a set A together with an operation $\sigma_{\mathbb{A}} : A^n \to A$, for any n-ary operation in Σ. A Σ-algebra homomorphism from $\mathbb{A}$ to $\mathbb{B}$ is a function $f : A \to B$ such that $f \circ \sigma_{\mathbb{A}} = \sigma_{\mathbb{B}} \circ \prod_n f$. We denote by Σ-**Alg** the category of Σ-algebras and their homomorphisms.*

The construction of Σ-terms defines a functor T_Σ which is part of a monad $\mathbb{T}_\Sigma$ whose unit is given by the subset inclusion injection $\iota : X \hookrightarrow T_\Sigma X$ and whose multiplication is given by term substitution (see below). Moreover, for any set X, the set $T_\Sigma X$ carries a Σ-algebra structure that makes it the *free Σ-algebra*: that means that for any Σ-algebra $\mathbb{A} = (A, \{\sigma_{\mathbb{A}}\}_{\sigma \in \Sigma})$, any map $f : X \to A$ extends uniquely to a Σ-algebra homomorphism $\hat{f} : T_\Sigma X \to A$ such that $\hat{f} \circ \iota = f$. We call maps $\hat{\gamma}$, for $\gamma : X \to T_\Sigma X$, substitution maps. Since a Σ-term over X is just a tree with operations in Σ as nodes and elements in X as leaves, we can uniquely write any term t as an expression of the form $\theta(x_1, \ldots, x_n)$, for some[6] $\theta \in T_\Sigma(\boldsymbol{n})$ and $(x_1, \ldots, x_n) \in \times X^{\boldsymbol{n}}$.[7] Accordingly, we see that if $t = \theta(x_1, \ldots, x_n)$ and $s_i = \vartheta_i(\vec{y}_i)$, then $\theta(\vartheta_1(\vec{y}_1), \ldots, \vartheta_n(\vec{y}_n))$ is the result of applying the substitution $x_i \mapsto s_i$ to t. Notice that algebraic operations on T_Σ are isomorphic to elements in $\bigcup_n T_\Sigma(\boldsymbol{n})$ [9].

Operations tell us how effects are produced, but the behaviour of such effects is defined by equations.

Definition 7. *1) An* algebraic *or* equational theory *$\mathbb{T} = (\Sigma, E)$ associates to each set X the collection $T_\Sigma X$ of Σ-term over X together with a set $E \subseteq T_\Sigma X \times T_\Sigma X$ of equations between such terms.*

2) *A (Σ, E)-algebra is a Σ-algebra $\mathbb{A}$ such that for any $(t, s) \in A$, and for any map $\alpha : X \to A$, we have $\hat{\alpha}(t) = \hat{\alpha}(s)$. We denote by (Σ, E)-**Alg** the subcategory of Σ-**Alg** whose objects are (Σ, E)-algebras.*

For a theory $\mathbb{T} = (\Sigma, E)$, we write $\sim_E$ (or simply $\sim$ if unambiguous) for the least congruence relation on terms that is closed under term substitution and contains E. The *free $\mathbb{T}$-theory* over X, notation $F_{\mathbb{T}}X$, is the quotient of $T_\Sigma X$ by $\sim$. Indeed, $F_{\mathbb{T}}X$ is a Σ-algebra (define $\sigma_{F_{\mathbb{T}}}([t_1]_\sim, \ldots, [t_n]_\sim)$

[6]For $n \in \mathbb{N}$, we write $\boldsymbol{n}$ for the set $\{1, \ldots, n\}$

[7]The family $\{T_\Sigma(\boldsymbol{n})\}_{n \in \mathbb{N}}$ together with its associated notion of composition (given by tree substitution) forms the so-called *tree operad* [43].

as $[\sigma(t_1, \ldots, t_n)]_\sim$) which obviously satisfies equations in E. Moreover, for any (Σ, E)-algebra $\mathbb{A}$, any map $f : X \to A$ extends uniquely to a Σ-homomorphism $\tilde{f} : F_{\mathbb{T}} X \to A$ such that $\tilde{f} \circ [-]_\sim \circ \iota = f$, where $[-]_\sim : T_\Sigma X \to F_{\mathbb{T}} X$ maps each term t to its equivalence class $[t]_\sim$. When unambiguous, we will write $[t]$ in place of $[t]_\sim$.

The construction $F_{\mathbb{T}}$ gives a functor which is part of a monad $\mathbb{F}_{\mathbb{T}}$, called the *free theory monad* of $\mathbb{T}$. Is is well-known that the Eilenberg-Moore category of a free theory monad $\mathbb{F}_{\mathbb{T}}$ is isomorphic to the category of (Σ, E)-algebras.

Example 8. The following are examples of equational theory. For readability, we write $t \sim s$ in place of $(t, s) \in E$.

1) The theory of divergence has a single 0-ary operation and no equation. Its free theory monad gives the maybe monad.
2) The theory of nondeterminism consists of a single binary operation $\vee$ together with the usual join semilattice equations [44]. Its associated free theory monad gives the finitary non-empty powerset monad. If we drop the idempotency equation $x \vee x \sim x$, we obtain the theory of multisets [45] and the associated multiset monad. If we also drop commutativity, we obtain the theory of lists and the associated list monad $\mathbb{L}$.
3) The theory of probabilistic nondeterminism has binary operations $+_p$ indexed by rational numbers $0 \leq p \leq 1$ subject to the usual axioms of a barycentric algebra [46]:
$$x +_p x \sim x; \quad x +_1 y \sim x; \quad x +_p y \sim y +_{1-p} x;$$
$$x +_p (y +_q z) \sim (x +_{\frac{p}{p+(1-p)q}} y) +_{p+(1-p)q} z.$$
The free theory monad of this theory gives the finitary distribution monad. The theory of multi-distribution or indexed valuations (and their corresponding monads) [15], [39], [40] is obtained by dropping the idempotency axiom $x +_p x \sim x$.
4) Fixed a monoid W, the theory of the writer monad has a unary operation $\mathtt{out}_w$ for each $w \in W$, and equations
$$\mathtt{out}_1(x) \sim x \qquad \mathtt{out}_w(\mathtt{out}_v(x)) \sim \mathtt{out}_{w \cdot v}(x).$$
5) Let R be a set of states which we assume to have two states only, say $R = \{0, 1\}$ (so that our location can stores booleans only). The theory of the reader monad has a binary operation $\mathtt{rd}$ and equations:
$$\mathtt{rd}(x, x) \sim x \quad \mathtt{rd}(\mathtt{rd}(w, x), \mathtt{rd}(y, z)) \sim \mathtt{rd}(w, z).$$
The intended meaning of a programs of the form $\mathtt{rd}(\mathtt{t}, \mathtt{s})$ is to read the value stored in the location and continue as $\mathtt{t}$ if the value is 0, and as $\mathtt{s}$ otherwise.

C. Rewriting and Relations

We will extensively work with relations. We denote by $\mathbf{Rel}$ the category with sets as objects and binary relations as arrows. As it is customary, we use the notation $R : X \nrightarrow Y$ for a relation $R \subseteq X \times Y$, and write $\mathbf{Rel}(X, Y)$ for the collection of relations of type $X \nrightarrow Y$. Notice that the subset inclusion order endow $\mathbf{Rel}(X, Y)$ with a complete lattice structure. We denote by $I_X : X \nrightarrow X$ the identity relation, by $R; S : X \nrightarrow Z$ the composition of $R : X \nrightarrow Y$ and $S : Y \nrightarrow Z$, and by $R^- : Y \nrightarrow X$ the dual or transpose of $R : X \nrightarrow Y$. As usual, we write R^* for the reflexive and transitive closure of R. Also, remember that any function $f : X \to Y$ can be regarded as a relation via its graph. As the context will disambiguate the notation, we write $f : X \to Y$ for the function f regarded as a relation (hence as an arrow in $\mathbf{Rel}$) In particular, the composition $g \circ f$ in $\mathbf{Set}$ corresponds to $f; g$ in $\mathbf{Rel}$. Finally, we will extensively use the following shunting laws [47]:

$$R; f \subseteq S \iff R \subseteq S; f^- \quad f^-; P \subseteq S \iff P \subseteq f; S.$$

An *abstract rewriting system* (ARS) [48] is a pair $(A, R : A \nrightarrow A)$ consisting of a set A together with a binary relation on A. Given the possible rewriting sequences an object of an ARS may have, sometimes we will be interested in those obtained following a strategy, the latter being a way to specify which reduction one should perform. Accordingly, we say that a strategy for R is a relation $R^{\mathtt{st}} \subseteq R$. Notice that a strategy need not be deterministic. Finally, we say that two relations R, S on A $\diamond$-*commute*[8] (resp. *commute*) if $R^-; S \subseteq S; R^-$ (resp. $R^-; S \subseteq S^*; (R^*)^-$). A relation is $\diamond$ (resp. *confluent*) if if it $\diamond$-commutes (resp. commutes) with itself.

IV. A Relational Theory of Monadic Rewriting System

In this section, we introduce our relational theory of *monadic rewriting systems*. Fixed a monad $\mathbb{T} = (T, \eta, \mu)$, a *monadic abstract rewriting system* (MARS) may be thought as a set A together with a (rewriting) relation $R : A \nrightarrow TA$. Since such relations play an important role in this work, we give them a name.

Definition 9. *Given a monad $\mathbb{T}$, a $\mathbb{T}$-*corelation $R : X \nrightarrow Y$ *is a relation $R : X \nrightarrow TY$.*

$\mathbb{T}$-corelations are T-coalgebras in $\mathbf{Rel}$, and they are the dual notion of a $\mathbb{T}$-relation [24], [26]. Thus, we may define a MARS as a pair $(A, R : A \nrightarrow A)$. This definition, however, is too liberal to give a meaningful notion of rewriting system. In fact, as already remarked in Section II, our main object of investigation are rewriting sequences, and to define them we need to be able to compose $\mathbb{T}$-corelations.

Given a candidate MARS $(A, R : A \nrightarrow A)$, we are interested in studying its rewriting sequences; and to do so, we need a notion of composition between $\mathbb{T}$-corelations. Here, we define $\mathbb{T}$-corelation composition relying on maps Γ lifting relations $R : X \nrightarrow TY$ to $\Gamma R : TX \nrightarrow TY$, which we then use to define the composition $R \cdot S : X \nrightarrow Z$ of two $\mathbb{T}$-corelations $R : X \nrightarrow Y$ and $S : Y \nrightarrow Z$ as $R; \Gamma S$.

Maps $\Gamma : \mathbf{Rel}(X, TY) \to \mathbf{Rel}(TX, TY)$ that guarantee good properties of $\mathbb{T}$-corelation composition turn out to coincide with the so-called *relational extensions* of $\mathbb{T}$ [24]–[27], a notion playing an important role in topology [24], [26], [27], coalgebra [25], [28], [29], logic [30], [31], and programming

[8] Read "diamond commute".

language theory [32]–[37]. This correspondence allows us to develop our relational theory of monadic rewriting systems relying on a vast body of results and techniques.

Definition 10 ([24])**.** *A* relational extension *of a monad* $\mathbb{T} = (T, \eta, \mu)$ *is a family of* monotone *maps* $\Gamma : \mathbf{Rel}(X, Y) \to \mathbf{Rel}(TX, TY)$ *such that:*

$$I = \Gamma(I) \quad \Gamma R; \Gamma S = \Gamma(R; S) \quad Ff = \Gamma f \quad (\Gamma R)^{-} = \Gamma(R^{-})$$
$$R; \eta_Y = \eta_X; \Gamma R \qquad \Gamma\Gamma R; \mu_Y = \mu_X; \Gamma R$$

If we ignore the last two equalities, we obtain the notion of a relational extension of the functor T*.*

A relational extension Γ of T is a way to extend T to a 2-functor on $\mathbf{Rel}$ that behaves as T on sets and functions, and as Γ on relations. Here, being a 2-functor means that $R \subseteq S$ implies $\Gamma R \subseteq \Gamma S$. If Γ is a relational extension of the monad $\mathbb{T}$, then Γ is a relational extension of T making the unit and multiplication of $\mathbb{T}$ natural. In particular, given a relational extension Γ, the map $\Gamma -; \mu : \mathbf{Rel}(X, TY) \to \mathbf{Rel}(TX, TY)$ gives the lifting we need to compose $\mathbb{T}$-corelations.

Lemma 11. *Given a relational extension* Γ *of* $\mathbb{T}$*, let* Δ *be* $\Gamma -; \mu$*. Then, the following hold, for all* $f : X \to TY$*,* $R : X \nrightarrow TY$*,* $S : Y \nrightarrow TZ$*.*

$$\Delta f = f^{\dagger} \qquad \eta; \Delta R = R \qquad \Delta(R; \Delta S) = \Delta R; \Delta S.$$

Now that we have the notion of a relational extension, we can give the following definition of a monadic rewriting system.

Definition 12. *Given a monad* $\mathbb{T}$ *with relational extension* Γ*, a* $(\mathbb{T}, \Gamma)$*-rewriting system is a pair* $(A, R : A \nrightarrow A)$*.*

Example 13. 1) Let us consider the monoid $(\mathbb{N}, +, 0)$ and instantiate the writer monad $\mathbb{W}$ on it. Define the relational extension Γ by $(n, x)\, \Gamma R\, (m, y)$ iff $n = m$ and $x\, R\, y$. As a consequence, the map $\Gamma -; \mu : \mathbf{Rel}(X, \mathcal{W}Y) \to \mathbf{Rel}(\mathcal{W}X, \mathcal{W}Y)$ gives $(n, x)\ (\Gamma R; \mu)\ (m, y)$ iff there exists p such that $m = n + p$ and $x\ R\ (p, y)$. Let us now consider the system defined by the $\mathbb{W}$-corelation $a\, R\, (a, 1)$. Then, we see that for any $n \geq 0$, we have the reduction sequence $a R^n (a, n)$. Here, R^n denotes the n-th $\mathbb{T}$-corelation composition of R with itself. Proposition 15 below ensures that such a composition is associative, and thus that the expression R^n is meaningful. For instance, for $n = 2$, we have the reduction $a\ (R \cdot R)\ (a, 2)$, i.e. $a\ (R; \Gamma R; \mu)\ (a, 2)$. Notice also that for $n = 0$, we have $R^0 = \eta$, which indeed gives $a\ R^0\ (a, 0)$.

2) Let us consider the list monad $\mathbb{L}$. Define the relational extension Γ by $xs \Gamma R ys$ if and only if $xs = [x_1, \ldots, x_n]$, $ys = [y_1, \ldots, y_n]$, and $x_i\ R\ y_i$, for any i. As a consequence, for any $R : X \to \mathcal{L}Y$ we have $xs\ (\Gamma R; \mu)\ ys$ iff there exist $ys_1, \ldots, ys_n$ such that $xs = [x_1, \ldots, x_n]$; $x_i\ R\ ys_i$ for any i; and $ys = ys_1 :: \cdots :: ys_n$, where $::$ denotes list concatenation. Now, consider the system defined by the $\mathbb{L}$-corelation $a R [a, a]$. Then, for any $n \geq 0$, we have the reduction sequence $a R^n [a, \ldots, a]$, where the length of $[a, \ldots, a]$ is 2^n. Notice that for $n = 0$, we have $a\ R^0\ [a]$.

MARSs are thus based on two parameters: a monad $\mathbb{T}$ specifying the (computational) effects produced during the rewriting process, and a relational extension Γ of $\mathbb{T}$ which we use to specify how to compose rewriting steps. Accordingly, it is natural to ask whether relational extensions exist in general. In the next section, we will answer this question. For the moment, we simply notice that we already gave a partial answer to such a question in Section II, where we showed that relational extensions of some monads do not exist. Actually, we have seen something different, namely that lifting operations satisfying compatibility and preservation do not exist. The following result connects these perspectives by showing that any relational extension gives a lifting satisfying compatibility and preservation. It follows that if there is no lifting satisfying compatibility and preservation, then there is no relational extension as well.

Proposition 14. *Let* $\mathbb{T}$ *be a monad with a relational extension* Γ *and let* $\Delta = \Gamma -; \mu$*. Then, for any* $R : X \nrightarrow Y$ *and algebraic operation* $f : T^n \Rightarrow T$*, we have:*

$$\prod_i \Delta R; f \subseteq f; \Delta R \qquad \eta; \Delta R = R.$$

Proposition 14 tells us that the lifting $\Gamma -; \mu$ induced by a relational extension Γ always satisfies compatibility and preservation, this way witnessing that our notion of a MARS is coherent with our informal notion of operational meaningfulness. Additionally, our notion of a MARS is also mathematically well-behaved, in the sense that the $\mathbb{T}$-corelation composition operation induced by a relational extension (i.e. $R \cdot S = R; \Gamma S; \mu$) indeed behaves as a notion of relation composition.

Proposition 15. *Let* Γ *be a relational extension of a* functor T *which is part of a monad* $\mathbb{T} = (T, \eta, \mu)$*. Then,* Γ *is a relational extension of* $\mathbb{T}$ *iff the following laws hold, for all* $R : X \nrightarrow Y$*,* $S : Y \nrightarrow Z$*, and* $P : Z \nrightarrow W$*:*

$$R \cdot \eta_Y = R \quad \eta_X \cdot S = S \quad R \cdot (S \cdot P) = (R \cdot S) \cdot P.$$

Notice that from Proposition 14 and Proposition 15 follows (by contraposition) Proposition 1 of Section II.

A. The Art and Craft of Relational Extensions

MARSs are defined with respect to two parameters: a monad $\mathbb{T}$ and a relational extension Γ of $\mathbb{T}$. However, we would like the monad $\mathbb{T}$ alone to determine its associated class of rewriting systems: that amounts to let $\mathbb{T}$ determine Γ. In 1970, Barr defined a canonical candidate relational extension $\hat{T}$ of a monad $\mathbb{T}$ and give necessary and sufficient conditions on $\mathbb{T}$ that make $\hat{T}$ a relational extension of $\mathbb{T}$. These conditions precisely isolate the class of monads that support monadic rewriting.

Definition 16. *Given a monad* $\mathbb{T} = (T, \eta, \mu)$ *and a relation* $R : X \nrightarrow Y$*, we can regard* R *as a set* $\underline{R} \subseteq X \times Y$*. In particular, the projections* $\pi_1 : \underline{R} \to X$*,* $\pi_2 : \underline{R} \to Y$ *give* $R =$

$\pi_1^-;\pi_2$. *The Barr extension* $\hat{T} : \mathbf{Rel}(X,Y) \to \mathbf{Rel}(TX,TY)$ *of* $\mathbb{T}$ *is defined as* $\hat{T}R = (T\pi_1)^-;T\pi_2$. *Elementwise, we have:*

$$\varphi\,\hat{T}R\,\psi \iff \exists\Phi \in T\underline{R}.\ T\pi_1(\Phi) = \varphi_1 \wedge T\pi_2(\Phi) = \psi.$$

Example 17. Let $R : X \nrightarrow Y$.

1) For the powerset monad, we have $u\hat{\mathcal{P}}Rv$ iff $\forall x \in u.\exists y \in v.\ x\,R\,y$ and $\forall y \in v.\exists x \in u.\ x\,R\,y$.
2) For the distribution monad, we have $\varphi\,\hat{\mathcal{D}}R\,\psi$ iff there exists $\Phi \in \mathcal{D}(X \times Y)$ such that $\sum_y \Phi(x,y) = \varphi(x)$, $\sum_x \Phi(x,y) = \psi(y)$, and $\Phi(x,y) > 0 \implies x\,R\,y$.
3) The Barr extensions for the writer and list monad are precisely the relational extensions given in Example 13.

The Barr extension of a monad $\mathbb{T} = (T,\eta,\mu)$ is not a relational extension of $\mathbb{T}$, in general: actually, it is not even a relational extension of T. Barr identified necessary and sufficient conditions on monads that guarantee their associated Barr extensions to be relational extensions. As a consequence, monads satisfying such conditions are precisely those monads that support monadic rewriting: they have a well-behaved notion of $\mathbb{T}$-corelation composition (which allows us to form rewriting sequences) and their Barr extension satisfies compatibility and preservation. Additionally, as we are going to see, such monads come with a number of powerful proof techniques for reasoning about their associated MARSs.

Definition 18 ([26])**.** *Consider functions* $X \xrightarrow{f} Z \xleftarrow{g} Y$ *and* $X \xleftarrow{p} W \xrightarrow{q} Y$.

1) *The commutative square* $p;f = q;g$ *is a* Beck-Chevalley square *(BC-square, for short) if* $p^-;q = f;g^-$*, or equivalently if* $q^-;p = g;f^-$.
2) *A functor* T *satisfies the* Beck-Chevalley condition *(BC, for short),*[9] *if it sends BC-squares to BC-squares: that is, for any square* $p;f = q;g$*,*

$$p^-;q = f;g^- \implies (Tp)^-;Tq = Tf;(Tg)^-$$

3) *A natural transformation* $\alpha : T \Rightarrow S$ *satisfies BC if its naturality diagrams*[10] $Sf;\alpha_Y = \alpha_X;Tf$ *are BC-squares. That is, if*

$$(Sf)^-;\alpha_X = \alpha_Y;(Tf)^- \text{ (or dually, } \alpha_X^-;Sf = Tf;\alpha_Y^-.)$$

Theorem 19 (Barr [24])**.** *Let* $\mathbb{T} = (T,\eta,\mu)$ *be a monad.*

1) *If* T *satisfies BC, then* $\hat{T}$ *is a relational extension of* T*. Moreover, such an extension is unique.*
2) *If* η *and* μ *satisfy BC, then* $\hat{T}$ *is a relational extension of* $\mathbb{T}$*. Moreover, such an extension is unique.*

Theorem 19 tells us that to specify a monadic rewriting system, it is sufficient to give a monad $\mathbb{T}$ and a pair $(A, R : A \nrightarrow A)$: the monad $\mathbb{T}$ itself gives the required relational extension as its Barr extension. Moreover, the structure of $\mathbb{T}$ alone determines whether $\hat{T}$ is a relational extension of $\mathbb{T}$, and thus whether rewriting works fine. Additionally, if $\mathbb{T}$ does not have the right structure, not only monadic rewriting via $\hat{T}$ is not possible, but no monadic rewriting on $\mathbb{T}$ is possible at all.

Structures satisfying property BC are called *weakly cartesian*, and thus we refer to monads whose underlying functor, unit, and multiplication all satisfy BC as *weakly cartesian monads* [17]–[20]. Examples of weakly cartesian monads include the the maybe, writer, list, multiset, and multi-distribution monads. Non-examples include the powerset, distribution, and reader monads (but also other monads not mentioned in this work, such as the the global state and group monads). This is coherent with the negative results we have seen in Section II.

Remark 20 (A Distributive Law Perspective)**.** In Section II, we have seen that no operationally meaningful monadic rewriting system can be defined upon the distribution monad. Another way to see that negative result is by means of non-existence of distributive laws [38]. Recall that, given monads $\mathbb{T} = (T,\eta^{\mathbb{T}},\mu^{\mathbb{T}})$ and $\mathbb{S} = (S,\eta^{\mathbb{S}},\mu^{\mathbb{S}})$, a distributive law of type $\mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ is a natural transformation $\delta : TS \implies ST$ satisfying suitable equations. Whenever we have a distributive law of type $\mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ we say that $\mathbb{T}$ distributes over $\mathbb{S}$.

Beck [38] proved that distributive laws of type $\mathbb{T}\mathbb{S} \Rightarrow \mathbb{S}\mathbb{T}$ are in a one-to-one correspondence with liftings of $\mathbb{S}$ to $\mathbf{EM}(\mathbb{T})$ and with extensions of $\mathbb{T}$ to $\mathbf{Kl}(\mathbb{S})$. Since $\mathbf{Rel}$ is nothing but $\mathbf{Kl}(\mathbb{P})$, we see that extensions of $\mathbb{T}$ to $\mathbf{Kl}(\mathbb{P})$ are exactly relational extensions of $\mathbb{T}$, and thus that there is a one-to-one correspondence between relational extensions and distributive laws of type $\mathbb{T}\mathbb{P} \Rightarrow \mathbb{P}\mathbb{T}$. As a consequence, a monad $\mathbb{T}$ has a relational extension if and only if there exists a distributive law of type $\mathbb{T}\mathbb{P} \Rightarrow \mathbb{P}\mathbb{T}$. The impossibility of (well-behaved) monadic rewriting on the distribution monad thus reflects the well-known result [39], [40] that the distribution monad does not distribute over the powerset monad. Since the same results holds also for, e.g., the powerset and group monads [41], we cannot have (well-behaved) monadic rewriting on these monads too.

From a distributive law perspective, we recover the Barr extension of a relation (regarded as a function) $R : X \to \mathcal{P}Y$ as $\hat{T}\ni \circ TR$, where $\hat{T}\ni : T\mathcal{P} \Rightarrow \mathcal{P}T$ is the candidate distributive law (known as the *power law* [49]) obtained as the Barr extension of the transpose $\ni : \mathcal{P}X \nrightarrow X$ of the set-membership relation $\in$. The counterpart of Theorem 19 states that $\hat{T}\ni$ gives a natural transformation whenever T satisfies BC, and a distributive law whenever $\mathbb{T}$ is weakly cartesian.

B. Monads, Points, and Diamonds

In this section, we witness the effectivness of our theory of monadic rewriting by developing two proof techniques for the analysis of MARSs. We will see a nontrivial application of such techniques in Section V, where we instantiate our general theory to study the λ-calculus with algebraic effects. In general, our techniques state that specific properties of $\mathbb{T}$-corelations are preserved by liftings. Here, the properties we are interested in are meant to be used to prove standardisation-like and confluence results. In the latter case, in particular, our

[9] Notice that T satisfies BC if and only if it preserves weak pullback diagrams [26, Proposition III.1.11.3].

[10] Where $SY \xrightarrow{\alpha_Y} TY \xleftarrow{Tf} TX$ and $SY \xleftarrow{Sf} SX \xrightarrow{\alpha_X} TX$.

main result (Theorem 22) states that the diamond property is preserved by the Barr extension construction. Let us expand on that.

For ease of exposition, let us introduce the following notation: for a monad $\mathbb{T} = (T, \eta, \mu)$, we write $\widetilde{T}$ for $\hat{T}-;\mu$. Now, given a MARS $(A, R : A \nrightarrow TA)$, we can study its behaviour by looking at the ARS $(TA, \widetilde{T}R : TA \nrightarrow TA)$ and regarding elements a in A as elements in TA via η. We know that if $\eta;\widetilde{T}R = R$ holds, then the R-rewriting behaviour of a coincides with the $\widetilde{T}R$-rewriting behaviour of $\eta(a)$. We also know (Proposition 15 and Appendix A) that the general law $\eta;\widetilde{T}R = R$ is equivalent to the law $S;\eta = \eta;\hat{T}S$ (where $S : X \nrightarrow Y$). But what is the advantage of studying $(TA, \widetilde{T}R)$ in place of (A, R)? The answer is simple: $(TA, \widetilde{T}R)$ being an ARS, we can rely on a vast body of results and techniques to analyse it. The drawback of this approach is that applying such results and techniques may be difficult, due to the intrinsic complexity of elements in TA. To be more concrete (cf. Section V), let us assume A is some set of inductively-defined terms (e.g. a set of Σ-terms or λ-terms), and that we want to prove a property of $(TA, \widetilde{T}R)$. Assume also that the theory of ARSs tells us that such a property is entailed by the diamond property, so that it is enough to prove that $(TA, \widetilde{T}R)$ has the latter property. At this point, we might want to rely on the inductive nature of A and to proceed by induction: but that is not *directly* possible, as the rewritten objects are elements in TA.

To overcome this difficulty, we introduce a family of proof techniques — which we dub *pointed techniques* — that allow us to prove specific properties of $(TA, \widetilde{T}R)$ by showing *pointed* version of such properties on (A, R). Here, we give two pointed techniques: one for proving standardisation-like results, and one stating that diamonds are preserved by $\widetilde{T}$. The latter, in particular, is nontrivial. To have a taste of such a technique, we consider the case of the probabilistic λ-calculus, where we write $\rightarrow$ for the probabilistic (β-like) reduction and $\Rightarrow$ for $\widetilde{\mathcal{D}}(\rightarrow)$. Our technique states that to prove the (lifted) diamond $\Leftarrow;\Rightarrow \subseteq \Rightarrow;\Leftarrow$ it is sufficient to prove its pointed version $\leftarrow;\rightarrow \subseteq \Rightarrow;\Leftarrow$. Notice that to prove the latter, we can indeed rely, e.g., on induction over λ-terms. Remarkably, the soundness of the pointed diamond technique crucially relies on the law $\hat{T}\hat{T}R;\mu = \mu;\hat{T}R$, so that we see that the same axioms ensuring monadic rewriting to be meaningful also guarantee the correctness of powerful techniques to study MARSs.

In the following, let $\mathbb{T} = (T, \eta, \mu)$ be a *weakly cartesian* monad. The first pointed technique we introduce states that factorisations are preserved by $\widetilde{T}$.

Proposition 21. *Given $R, P : X \nrightarrow Y$, $S, Q : Y \nrightarrow Z$, we have:*

$$R;\widetilde{T}S \subseteq P;\widetilde{T}Q \implies \widetilde{T}R;\widetilde{T}S \subseteq \widetilde{T}P;\widetilde{T}Q.$$

Proof sketch. By symbolic manipulations relying on $\hat{T}\hat{T}R;\mu = \mu;\hat{T}R$. □

Next, we move to our main result.

Theorem 22. *The pointed diamond technique*

$$R^-;S \subseteq \widetilde{T}S;(\widetilde{T}R)^- \implies (\widetilde{T}R)^-;\widetilde{T}S \subseteq \widetilde{T}S;(\widetilde{T}R)^-$$

is sound.

The proof of Theorem 22 is nontrivial and crucially relies on weak cartesianess of $\mathbb{T}$ (of μ, actually). We give a complete proof in Appendix A.

C. Back to Algebra

So far, we focused on monadic rewriting in the abstract, i.e. with respect to arbitrary monads. When monads are given via equational theories, as it is in the case of algebraic effects, we can take advantage of that information to give syntactic-like characterisations of some of the notions introduced in previous sections. Here, we give such characterisations for the Barr extension of free theory monads. Furthermore, we identify a sufficient condition on the shape of an equational theory $\mathbb{T}$ that guarantees the Barr extension of the free theory monad of $\mathbb{T}$ to be a relational extension.

Let $\mathbb{T} = (\Sigma, E)$ be an algebraic theory. We first spell out the definition of the Barr extension of $\mathbb{F}_{\mathbb{T}}$, the free theory monad of $\mathbb{T}$. Actually, since to define $\mathbb{F}_{\mathbb{T}}$-corelation composition we have to lift relations in $\mathbf{Rel}(X, F_{\mathbb{T}}Y)$ to relations in $\mathbf{Rel}(F_{\mathbb{T}}X, F_{\mathbb{T}}Y)$, we directly work with the map $\widetilde{F_{\mathbb{T}}} : \mathbf{Rel}(X, F_{\mathbb{T}}Y) \rightarrow \mathbf{Rel}(F_{\mathbb{T}}X, F_{\mathbb{T}}Y)$ given in previous section: $\widetilde{F_{\mathbb{T}}}R = \hat{F_{\mathbb{T}}}R;\mu$. Explicitly, given a relation $R : X \nrightarrow F_{\mathbb{T}}Y$, we have:

$$[t]\ \widetilde{F_{\mathbb{T}}}R\ [s] \iff \exists n.\exists \vartheta \in T_{\Sigma}(\boldsymbol{n}). \begin{cases} t \sim \vartheta(x_1, \ldots, x_n) \\ s \sim \vartheta([s_1], \ldots, [s_n]) \\ \forall i.\ x_i\ R\ [s_i]. \end{cases}$$

Example 23. Let $\mathbb{T} = (\Sigma, E)$ be an equational theory and fix a $F_{\mathbb{T}}$-corelation $R : X \nrightarrow Y$.

1) For $\mathbb{T}$ the theory of barycentric algebras, we have $[t]\ \widetilde{F_{\mathbb{T}}}R\ [s]$ if and only if $t \sim \sum_{i=1}^{n} p_i{:}x_i$, $s \sim \sum_{i=1}^{n} p_i{:}[s_i]$, and $x_i\ R\ [s_i]$, for any $i \leq n$.
2) For $\mathbb{T}$ the theory of join semilattices, we have $[t]\ \widetilde{F_{\mathbb{T}}}R\ [s]$ if and only if $t \sim \bigvee_{i=1}^{n} x_i$, $s \sim \bigvee_{i=1}^{n} [s_i]$, and $x_i\ R\ [s_i]$, for any $i \leq n$.
3) For $\mathbb{T}$ the theory of the writer monad (on a monoid W), we have $[t]\ \widetilde{F_{\mathbb{T}}}R\ [s]$ if and only if $t \sim \mathtt{out}_w(x)$, $s \sim \mathtt{out}_w([s'])$, and $x\ R\ [s']$, where $w \in W$.

Let us now think about relations as arrows in $\mathbf{Kl}(\mathbb{P})$. That is, we regard a relation $R : X \nrightarrow Y$ as a function $R : X \rightarrow \mathcal{P}Y$. Given a monad $\mathbb{T}$, recall the power law $\delta = \hat{T}\ni$ of Remark 20. The latter induces a map $\mathcal{P}_\delta : \Sigma\text{-}\mathbf{Alg} \rightarrow \Sigma\text{-}\mathbf{Alg}$ sending a Σ-algebra $\mathbb{A} = (A, \{\sigma_{\mathbb{A}}\}_{\sigma\in\Sigma})$ to the Σ-algebra $\mathcal{P}_\delta\mathbb{A} = (\mathcal{P}A, \{\delta \circ \sigma_{\mathbb{A}}\}_{\sigma\in\Sigma})$. In particular, we have $(\delta \circ \sigma_{\mathbb{A}})(u_1, \ldots, u_n) = \{\sigma_{\mathbb{A}}(a_1, \ldots, a_n) \mid a_i \in u_i\}$, so that we see that $\mathcal{P}_\delta\mathbb{A}$ corresponds to the so-called *power* or *complex* algebra on $\mathbb{A}$ [21], [22]. Since, in particular, $\mathcal{P}F_{\mathbb{T}}$ builds Σ-algebras, we can take advantage of the free algebra property of T_Σ and define a syntactic notion of lifting that allows us

to characterise $\hat{F_{\mathbb{T}}}$ syntactically, provided that the power law is a distributive law.

Definition 24. *Given a theory $\mathbb{T} = (\Sigma, E)$, we define the* syntactic extension $R^{\Sigma} : T_{\Sigma}X \nrightarrow F_{\Sigma}Y$ *of $R : X \nrightarrow F_{\Sigma}Y$ as the free algebra homomorphism of R (regarded as a function of type $X \to \mathcal{P}F_{\Sigma}Y$). Equivalently, we inductively define R^{Σ} as follows:*

$$\frac{x\ R\ [s]}{x\ R^{\Sigma}\ [s]} \quad \frac{t_1\ R^{\Sigma}\ [s_1] \cdots t_n\ R^{\Sigma}\ [s_n] \quad \sigma \in \Sigma_n}{\sigma(t_1, \ldots, t_n)\ R^{\Sigma}\ [\sigma(s_1, \ldots, s_n)]}$$

Theorem 25. *Let $\mathbb{T} = (\Sigma, E)$ be a theory. If the power law is a distributive law, then for any $\mathbb{F}_{\mathbb{T}}$-corelation $R : X \nrightarrow Y$ we have: $[t]\ \widetilde{F_{\mathbb{T}}}R\ [s]$ iff $t\ R^{\Sigma}\ [s]$.*

Proof sketch. Since $\hat{F_{\mathbb{T}}}(\ni)$ is a distributive law, $\mathcal{P}^{\delta}$ extends from Σ-**Alg** to (Σ, E)-**Alg** (recall that (Σ, E)-**Alg** $\cong$ $\mathbf{EM}(\mathbb{F}_{\mathbb{T}})$). We then exploit the free algebra properties of $T_{\Sigma}X$ and $F_{\mathbb{T}}X$. □

Theorem 25 allows one to reason about the syntactic structure of terms. In fact, it states that we can perform rewriting directly on terms in $T_{\Sigma}X$, rather than on their equivalence classes: whenever $t \sim s$ and t rewrites into $[t']$, then s rewrites into $[s']$, for some $[s']$ such that $t' \sim s'$ (and thus $[t'] = [s']$).

In particular, if $\mathbb{F}_{\mathbb{T}}$ is weakly cartesian, then the hypothesis of Theorem 25 holds. The notion of a weakly cartesian monad, however, is rather abstract and it does not take advantage of working with algebraic theories. This raises a natural question: are there some conditions on the shape of a theory that guarantee good properties of the corresponding monadic rewriting? The crucial passage to prove Theorem 25 is to show that the power algebra construction $\mathcal{P}^{\delta} : \Sigma\text{-}\mathbf{Alg} \to \Sigma\text{-}\mathbf{Alg}$ lifts to (Σ, E)-**Alg**. This is equivalent to show that δ is a distributive law, and thus that $\hat{F_{\mathbb{T}}}$ is a relational extension of $\mathbb{F}_{\mathbb{T}}$. Therefore, what we need is to find conditions on $\mathbb{T}$ making $\mathcal{P}^{\delta}$ lift to (Σ, E)-**Alg**: but that means nothing but finding conditions ensuring that $\mathcal{P}^{\delta}$ preserves equations in E. The whole issue can be thus rephrased as follows: when the power algebra of a (Σ, E)-algebra satisfies E? This question has been answered in the late 50s by Gautam [21], who showed that if all equations in E are *linear*, then they are preserved by the power algebra construction. Notice that all the 'problematic' equations met in Section II are *non-linear*, and that the result by Gautam perfectly fits with our general results on arbitrary monads, as the free theory monad of a linear theory is weakly cartesian [50].

Theorem 26 (Gautam [21])**.** *Let $\mathbb{T} = (\Sigma, E)$ be an equational theory. Recall that an equation $(t, s) \in E$ is* linear *if t and s have the same variables and each of them appears exactly once (in each term). If $\mathbb{T}$ is linear (i.e. all equations in E are linear), then $(\mathcal{P}F_{\mathbb{T}}X, \{\hat{F_{\mathbb{T}}}(\ni) \circ \sigma_{F_{\mathbb{T}}}\}_{\sigma \in \Sigma})$ is a (Σ, E)-algebra, for any set X.*

Corollary 27. *Let $\mathbb{T} = (\Sigma, E)$ be* linear *equational theory. Then $\hat{F_{\mathbb{T}}}$ is a relational extension of $\mathbb{F}_{\mathbb{T}}$. Moreover, for any $\mathbb{F}_{\mathbb{T}}$-corelation $R : X \nrightarrow Y$ we have: $[t]\ \widetilde{F_{\mathbb{T}}}R\ [s]$ iff $t\ R^{\Sigma}\ [s]$.*

Example 28. Obviously, the idempotency equations $x \vee x \sim x$ and $x +_p x \sim x$ are not linear, as it is not the inverse law $x + (-x) \sim 0$. Therefore, the theories of the distribution, powerset, and group monad are not linear, as it is not the theory of the reader monad (but see Remark 46) The theories of the maybe, writer, (finite and non-empty) multi-set and list, multi-distribution, and monoid monads are linear.

V. Call-by-Value λ-calculus with Algebraic Effects

Now that we have set up an abstract machinery for monadic rewriting, we instantiate it to study the operational theory of λ-calculi extended with algebraic operations. More specifically, we study extensions of the call-by-value λ-calculus [23] with operations from a theory $\mathbb{T} = (\Sigma, E)$. From now on, let $\mathbb{T} = (\Sigma, E)$ be an arbitrary linear algebraic theory.

Definition 29. *The calculus $\Lambda_{\Sigma}^{\mathsf{cbv}}$ has terms and values defined by the following grammar, where $\sigma \in \Sigma$.*

$$\mathtt{t}, \mathtt{s} ::= \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t} \mid \mathtt{ts} \mid \sigma(\mathtt{t}, \ldots, \mathtt{t}) \qquad \mathtt{v}, \mathtt{w} ::= \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t}.$$

Operations in Σ act as effect-triggering operations, so that we may informally read the behaviour of a term $\sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ as 'perform the effect prescribed by σ, and then continue with one of the $\mathtt{t}_i$s'. For instance, by taking operations $\mathtt{out}_w$ of the theory of the writer monad, we obtain the call-by-value λ-calculus with output, whereas by taking operations $+_p$ of the theory of barycentric algebras, we obtain the probabilistic call-by-value λ-calculus [16], [51].

We adopt standard conventions [3]. In particular, we work with terms modulo renaming of bound variables, and denote by $\mathtt{t}\langle\mathtt{s}/\mathtt{x}\rangle$ the capture-avoiding substitution of $\mathtt{s}$ for the free occurrences of $\mathtt{x}$ in $\mathtt{t}$. Finally, we write Λ and $\mathcal{V}$ for the sets of terms and values, respectively.

In order to define notions of reduction, we define *contexts* and *surface contexts* [16], the latter being contexts not allowing reductions to happen under neither the scope of a λ nor of an operation.

$$\mathtt{C} ::= \Box \mid \lambda\mathtt{x}.\mathtt{C} \mid \mathtt{tC} \mid \mathtt{Ct} \mid \sigma(\mathtt{t}_1, \ldots, \mathtt{C}, \ldots, \mathtt{t}_n)$$
$$\mathtt{S} ::= \Box \mid \mathtt{tS} \mid \mathtt{St}.$$

As it is customary, for a context $\mathtt{C}$ we write $\mathtt{C}\langle\mathtt{t}\rangle$ for the substitution of $\Box$ with $\mathtt{t}$ in $\mathtt{C}$, allowing the capture of free variables

Remark 30. Notice that performing effectful reductions inside surface contexts means, besides not reducing under λ-abstraction, to always reduce the outermost operation symbol (hence performing the outermost effect). This restriction is crucial for our results (see Remark 33) and it is standard practice in calculi with algebraic effects [10].

a) Reductions: Having defined the syntax of $\Lambda_{\Sigma}^{\mathsf{cbv}}$, we now look at reduction rules for it. $\Lambda_{\Sigma}^{\mathsf{cbv}}$ has two kinds of reduction: β-reductions, and effectful reductions. As usual, a β-reduction performs a *pure* computational step, whereas an effectful reduction is basically a reduction on terms of the form

$\sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n)$, and it is the real effect producer. We begin with term-to-term reductions.

Definition 31. *One-step β- and σ_i-reduction relations $\mapsto_\beta, \mapsto_{\sigma_i} : \Lambda \nrightarrow \Lambda$ are thus defined:*

$$(\lambda \mathtt{x}.\mathtt{t})\mathtt{v} \mapsto_\beta \mathtt{t}\langle \mathtt{v}/\mathtt{x}\rangle; \qquad \sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n) \mapsto_{\sigma_i} \mathtt{t}_i.$$

One-step reductions do not provide any semantical information on the operation reduced (and hence on the effect produced). To overcome this problem, we move from term reductions to $\mathbb{F}_{\mathbb{T}}$-corelations.

Definition 32. *Define the relations $\to_\beta, \to_\sigma : \Lambda_\Sigma \nrightarrow F_{\mathbb{T}}\Lambda_\Sigma$ by closing $\mapsto_\beta$-reductions under* arbitrary *contexts, and $\mapsto_{\sigma_i}$ reductions under* surface *contexts:*

$$\mathtt{C}\langle(\lambda \mathtt{x}.\mathtt{t})\mathtt{v}\rangle \to_\beta [\mathtt{C}\langle \mathtt{t}\langle \mathtt{v}/\mathtt{x}\rangle\rangle]$$
$$\mathtt{S}\langle\sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n)\rangle \to_\sigma [\sigma(\mathtt{S}\langle \mathtt{t}_1\rangle, \ldots, \mathtt{S}\langle \mathtt{t}_n\rangle)].$$

We write $\to_\Sigma$ for $\bigcup_{\sigma\in\Sigma} \to_\sigma$, and $\to$ for $\to_\beta \cup \to_\Sigma$.

We define $\Rightarrow_r$: $F_{\mathbb{T}}\Lambda \nrightarrow F_{\mathbb{T}}\Lambda$ as $\widetilde{F_{\mathbb{T}}}(\to_r \cup \eta)$, where $r \in \{\beta, \sigma, \Sigma\}$. In particular, we have $\Rightarrow = \widetilde{F_{\mathbb{T}}}(\to \cup \eta)$. Please notice that the presence of the (function regarded as a) relation η ensures $\Rightarrow$ to be reflexive. Finally, we define the notion of a *surface reduction* $\xrightarrow{\mathtt{s}}$ by replacing the context $\mathtt{C}$ with $\mathtt{S}$ also for $\to_\beta$ in Definition 32. The relation $\overset{\mathtt{s}}{\Rightarrow}$ is defined accordingly. A reduction is *deep* (notation $\overset{\neg\mathtt{s}}{\to}$, $\overset{\neg\mathtt{s}}{\Rightarrow}$) if it is not surface. Please notice that surface recuction restricts β-reduction to surface contexts, this way forbidding β-reductions under the scope of a λ-abstraction, and thus making the calculus weak.

Remark 33. Effectful reduction is restricted to surface contexts, since otherwise one could obtain from a program outcomes which are irreversibly different, with no hope neither for standardisation nor for confluence to hold [16]. Intuitively, this is because evaluating an effectful subterm $\mathtt{t}$ (hence producing an effect) and *then* copying it, or *first* copying the subterm $\mathtt{t}$ and then evaluating each copy (this way producing the same effect, but several times) give different results. It is important to stress that, instead, β-reductions are *unconstrained*. That is, both $\to$ and $\Rightarrow$ reduce β-redexes inside *arbitrary* contexts. Thus, for instance, the probabilistic term $(\mathtt{Ix})(\mathtt{II} +_p (\Delta\Delta +_q \mathtt{I}\Delta))$ (where $\mathtt{I} = \lambda\mathtt{x}.\mathtt{x}$ and $\Delta = \lambda\mathtt{x}.\mathtt{xx}$) has four β-redexes (namely, $\mathtt{Ix}$, $\mathtt{II}$, $\Delta\Delta$, and $\mathtt{I}\Delta$), which can *all* be reduced by $\to$ (even if inside algebraic operations). The term also has two effectful redexes, namely $\mathtt{II} +_p (\Delta\Delta +_q \mathtt{I}\Delta)$ and $\Delta\Delta +_q \mathtt{I}\Delta$. However, since effectful reductions must be surface, only the first one can be reduced.

A. Surface Standardization, Confluence, and Diamonds

In this section, we study the extension of two fundamental results in the theory of (pure) λ-calculus to an effectful setting, namely a suitable form of standardisation, that we call *surface standardisation*, and *confluence*. We first prove surface standardisation: any reduction sequence $[t] \Rightarrow^* [s]$ can be factorised as a sequence of surface reductions followed by a sequence of non-surface reductions. This is analogous to a classical result in λ-calculus: a sequence of β-steps can be reorganised so as to first reducing head redexes and then everything else [3, Theorem. 11.4.6]. Remarkably, surface standardisation holds for *all* (linear) algebraic effects.

Next, we prove a confluence theorem stating that if effects are *commutative*, meaning that effects are independent of the order in which they are produced, then monadic reduction is confluent.

1) Surface standardisation: Let us begin with surface standardisation. Our goal is to prove that any reduction sequence $\Rightarrow^*$ can be factorised as $\overset{\mathtt{s}}{\Rightarrow}{}^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}^*$. We follow the the modular proof by Accattoli et al. [52] and rely on standardisation for the pure call-by-value λ-calculus.

Lemma 34 ([52])**.** *If $\Rightarrow_\beta^* \subseteq \overset{\mathtt{s}}{\Rightarrow}{}_\beta^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}_\beta^*$ and $\overset{\neg\mathtt{s}}{\Rightarrow}{}_\beta; \Rightarrow_\Sigma \subseteq \Rightarrow_\Sigma; \Rightarrow_\beta$, then $\Rightarrow^* \subseteq \overset{\mathtt{s}}{\Rightarrow}{}^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}^*$*

We now prove that both the assumptions in Lemma 34 hold, this way obtaining the desired standardisation result. Let us begin with the first one. First, notice that an analogous result holds for the pure call-by-value λ-calculus.

Theorem 35 ([23], [53])**.** *Let $\Lambda^{\mathtt{cbv}}$ be the pure call-by-value λ-calculus (i.e. our calculus instantiated with the empty theory, so that $\Rightarrow_\beta = \to_\beta$). Then $\Rightarrow_\beta^* \subseteq \overset{\mathtt{s}}{\Rightarrow}{}_\beta^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}_\beta^*$.*

We now lift Theorem 35 to $\Lambda_\Sigma^{\mathtt{cbv}}$. To do so, we define a translation $-^\lambda$ from $\Lambda_\Sigma^{\mathtt{cbv}}$ to $\Lambda^{\mathtt{cbv}}$ preserving β-reductions in *both* directions. To achieve such a goal, we need to ensure that whenever we have a β-reduction in $\mathtt{t}^\lambda$ (in $\Lambda^{\mathtt{cbv}}$), then the β-redex reduced is the $-^\lambda$ image of a β-redex in $\mathtt{t}$. Stated otherwise, we need $-^\lambda$ to preserve β-redexes back and forth, and thus to ensure that a Σ-redex is not mapped into a β-redex.

Definition 36. *For any operation $\sigma \in \Sigma$, let us fix two distinguished variables x_σ, y_σ. Define:*

$$\mathtt{x}^\lambda \triangleq \mathtt{x} \qquad (\lambda\mathtt{x}.\mathtt{t})^\lambda \triangleq \lambda\mathtt{x}.\mathtt{t}^\lambda$$
$$(\mathtt{st})^\lambda \triangleq \mathtt{s}^\lambda\mathtt{t}^\lambda \quad (\sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n))^\lambda \triangleq y_\sigma(\lambda x_\sigma.\mathtt{t}_1^\lambda)\cdots(\lambda x_\sigma.\mathtt{t}_n^\lambda).$$

Lemma 37. *Let $\xrightarrow{\mathtt{r}} \in \{\to_\beta, \xrightarrow{\mathtt{s}}_\beta\}$. Then: $\mathtt{t} \xrightarrow{\mathtt{r}} [\mathtt{s}]$ (in $\Lambda^{\mathtt{cbv}}$) if and only if $\mathtt{t}^\lambda \xrightarrow{\mathtt{r}} [\mathtt{s}^\lambda]$ (in $\Lambda^{\mathtt{cbv}}$).*

Proof sketch. The left to right direction of is straightforward. The right to left direction follows by very definition of translation, since $(\sigma(\mathtt{t}_1, \ldots, \mathtt{t}_n))^\lambda$ cannot be a β-redex. □

Theorem 38 (Surface Standardization)**.** $\Rightarrow^* \subseteq \overset{\mathtt{s}}{\Rightarrow}{}^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}^*$.

Proof Sketch. From Lemma 37 we infer $\Rightarrow_\beta^* \subseteq \overset{\mathtt{s}}{\Rightarrow}{}_\beta^*; \overset{\neg\mathtt{s}}{\Rightarrow}{}_\beta^*$. Moreover, since we have $\overset{\neg\mathtt{s}}{\to}_\beta; \Rightarrow_\sigma \subseteq \to_\sigma; \Rightarrow_\beta$, from Proposition 21 follows $\overset{\neg\mathtt{s}}{\Rightarrow}{}_\beta; \Rightarrow_\Sigma \subseteq \Rightarrow_\Sigma; \Rightarrow_\beta$.We thus conclude the thesis using Lemma 34. □

2) Confluence: It is now time to investigate confluence of $\Rightarrow$. We immediately notice that $\Rightarrow$ is not confluent, in general. To see why, we simply consider the term $\mathtt{t} =$

$(\mathtt{out}_w(\mathtt{x}))(\mathtt{out}_v(\mathtt{x}))$, which gives the following reduction sequences:

$$\mathtt{t} \to [\mathtt{out}_w(\mathtt{x}(\mathtt{out}_v(\mathtt{x})))] \Rightarrow [\mathtt{out}_{w\cdot v}(\mathtt{xx})]$$
$$\mathtt{t} \to [\mathtt{out}_v((\mathtt{out}_w(\mathtt{x}))\mathtt{x})] \Rightarrow [\mathtt{out}_{v\cdot w}(\mathtt{xx})]$$

Confluence cannot be achieved, as the outputted strings $w \cdot v$ and $v \cdot w$ witness the order in which reductions happened. Notice also that the reductions performed are actually surface.

Although providing a counterexample to confluence, the above example suggests conditions under which confluence may be recovered, namely the commutativity of the effects performed (i.e. $w \cdot v = v \cdot w$, in our example). Commutative effects are a well-known concept in programming language theory, as they support a number of program refactoring and optimisation techniques.[11] Working with equational theories, we can formalise the notion of a commutative effect as follows.

Definition 39. *Given $\theta \in T_\Sigma(\boldsymbol{n})$ and $\vartheta \in T_\Sigma(\boldsymbol{m})$, we say that θ commute with ϑ if:*

$$\theta(\vartheta(x_{1,1}, \ldots, x_{1,m}), \ldots, \vartheta(x_{n,1}, \ldots, x_{n,m})) \\ \sim \vartheta(\theta(x_{1,1}, \ldots, x_{n,1}), \ldots, \theta(x_{1,m}, \ldots, x_{n,m})).$$

A theory $\mathbb{T} = (\Sigma, E)$ is commutative *if θ commutes with ϑ, for all θ and ϑ.*

Example 40. The theory of the powerset, multiset, list, distribution, multi-distribution, partiality, and cost monad are commutative, whereas the theory of the writer monad is not.

Now that we have the notion of a commutative effect, we prove that if effects are commutative, then $\Rightarrow$ is confluent. Our proof builds upon Theorem 22, which allows us to reduce diamond properties to *pointed* diamond properties. In particular, the following result then gives confluence of *surface* reduction, in a strong form.

Lemma 41. *Let $\gamma, \delta \in \Sigma \cup \{\beta\}$, $\rightarrowtail = \widetilde{F_{\mathbb{T}}}(\to)$, and $\mathtt{t}$ be a term with two distinct* surface *redexes. If $[\theta(\mathtt{t}_1, \ldots, \mathtt{t}_n)] \,{}_\gamma\overset{\mathtt{s}}{\leftarrow}\, \mathtt{t} \,\overset{\mathtt{s}}{\to}_\delta\, [\vartheta(\mathtt{s}_1, \ldots, \mathtt{s}_m)]$ and θ commutes with ϑ, then $[\theta(\mathtt{t}_1, \ldots, \mathtt{t}_n)] \overset{\mathtt{s}}{\rightarrowtail}_\delta [p] \,{}_\gamma\overset{\mathtt{s}}{\leftarrowtail}\, [\vartheta(\mathtt{s}_1, \ldots, \mathtt{s}_m)]$.*

Proof sketch. By induction on $\mathtt{t}$ relaying on commutativity of θ and ϑ, and observing that since reductions are surface, $\mathtt{t}$ must be of the form $\mathtt{pq}$. □

From Lemma 41 and Theorem 22 it follows that surface reduction is diamond.

Proposition 42 (Surface is $\diamond$)**.** *If $\mathbb{T} = (\Sigma, E)$ is commutative, then $\overset{\mathtt{s}}{\Rightarrow}$ is diamond, and thus confluent. The same holds, in particular, for $\Rightarrow_\Sigma$.*

From the confluence of surface reduction, and the fact that $\to_\beta$ is confluent, we obtain confluence of $\Rightarrow$.

[11] For instance, in languages with sequencing commutativity essentially corresponds to the soundness of the program transformation $\mathbf{let}\ x = M\ \mathbf{in}\ (\mathbf{let}\ y = N\ \mathbf{in}\ L) \equiv \mathbf{let}\ y = N\ \mathbf{in}\ (\mathbf{let}\ x = M\ \mathbf{in}\ L)$, (with x not appearing free in N and y not appearing free in M).

Theorem 43 (Confluence)**.** *If $\mathbb{T} = (\Sigma, E)$ is commutative, then $\Rightarrow$ is confluent.*

Proof sketch. By Hindley-Rosen lemma [3, Proposition 3.3.5], to prove that $\Rightarrow\ =\ \Rightarrow_\beta \cup \Rightarrow_\Sigma$ is confluent it is enough to show that $\Rightarrow_\beta$ and $\Rightarrow_\Sigma$ are confluent and commute with each other. Commutation of $\Rightarrow_\beta$ and $\Rightarrow_\Sigma$ follows from Theorem 22 taking advantage of Lemma 41, opportunely completed with an easy analysis of the non-surface $\to_\beta$ steps. Confluence of $\Rightarrow_\beta$ follows from confluence of β-reduction in the pure call-by-value λ-calculus, by Lemma 37. □

VI. Conclusion

In this paper, we have introduced a relational theory of monadic rewriting systems and showed how such a theory can be fruitfully applied to the study of effectful computations. Our development goes through a number of theoretical results on monads and relational extensions, as well as on a family of negative results showing that having a well-behaved notion of (relational) monadic rewriting is just not possible on some monads. Moving from those negative results, we have identified a class of monads, namely the class of weakly cartesian monads, on which monadic rewriting works well. As a further evidence of that, we showed that the defining axioms of a weakly cartesian monad allow us to prove a number of powerful proof techniques (viz. Proposition 21 and Theorem 22), which we dub pointed techniques, for the analysis of monadic (abstract) rewriting systems. We have also showed that when monads are given in terms of equational theories, then a sufficient condition for having well-behaved rewriting is that all equations in the theory are linear. Last but not leas, we have witnessed the effectiveness of our theory by making an operational analysis of the call-by-value λ-calculus with algebraic effects, the latter being a prominent formalism for the study of higher-order effectful languages. In this respect, we have proved two main theorems: surface standardisation (Theorem 38) and confluence (Theorem 43). The former holds for all algebraic effects, whereas the latter hold for commutative effects only. Remarkably, both such results are proved relying on the aforementioned pointed techniques.

a) Future Work: This work is the first stage of a large research project aiming to develop a relational theory of monadic rewriting systems. As such, it opens several research directions. Among those, the authors are currently working on infinitary and asymptotic monadic rewriting [14]. In fact, when dealing with computational effects, there are interesting properties of computations that cannot be studied in terms of finitary rewriting sequences, examples of those being almost sure termination in probabilistic computation, stream analysis in computations with output, and divergence in pure computations. Although at a preliminary stage, the results obtained in this direction are encouraging.

b) Related Work: Categorical approaches to rewriting based on monads are not new [54]–[56]. However, to the best of the authors' knowledge all such approaches are concerned

with *pure* rewriting and use monads to structure objects of ARSs, this way generalising the (algebraic) signatures one has in term rewriting systems [1]. Among these approaches, the work by Fritz and Perrone [56] uses algebras over a monad to model *pure* partial evaluation functions, whereby only specific components of complex expressions are evaluated. Although partial evaluations seem not to share similarities with our notion of effectful rewriting, it is worth noticing that weakly cartesian monads guarantee good properties (e.g. transitivity) of such maps.

Effectful rewriting systems have been studied in isolation, focusing on single specific effects only, with a growing interest in probabilsitic rewriting [13]–[15]. The situation is similar if one looks at rewriting systems for effectful λ-calculi, where the rewriting theory of several specific effectful calculi has been studied [16], [51], [57]. Operational properties of calculi with arbitrary algebraic effects have been studied by Plotkin and Power [10], who gave operational semantics to a PCF-like language extended with algebraic operations. Such a semantics (and the like, e.g. [58], [59]), however, is given by a fixed deterministic reduction strategy, and therefore does not allow for a full operational analysis of algebraic effects.

References

[1] Terese, *Term rewriting systems*, ser. Cambridge tracts in theoretical computer science. Cambridge University Press, 2003, vol. 55.

[2] M. H. A. Newman, "On theories with a combinatorial definition of "equivalence"," *Annals of Mathematics*, vol. 43, no. 2, pp. 223–243, 1942.

[3] H. Barendregt, *The lambda calculus: its syntax and semantics*, ser. Studies in logic and the foundations of mathematics. North-Holland, 1984.

[4] X. Leroy, "The ZINC experiment: an economical implementation of the ML language," INRIA, Technical report 117, 1990.

[5] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, ser. Addison-Wesley series in computer science / World student series edition. Addison-Wesley, 1986.

[6] E. Moggi, "Computational lambda-calculus and monads," in *Proc. of LICS 1989*. IEEE Computer Society, 1989, pp. 14–23.

[7] ——, "Notions of computation and monads," *Inf. Comput.*, vol. 93, no. 1, pp. 55–92, 1991.

[8] S. MacLane, *Categories for the Working Mathematician*. Springer-Verlag, 1971.

[9] G. D. Plotkin and J. Power, "Algebraic operations and generic effects," *Applied Categorical Structures*, vol. 11, no. 1, pp. 69–94, 2003.

[10] ——, "Adequacy for algebraic effects," in *Proc. of FOSSACS 2001*, 2001, pp. 1–24.

[11] ——, "Semantics for algebraic operations," *Electr. Notes Theor. Comput. Sci.*, vol. 45, pp. 332–345, 2001.

[12] O. Bournez and C. Kirchner, "Probabilistic rewrite strategies. applications to ELAN," in *Proc. of RTA 2002*, ser. Lecture Notes in Computer Science, S. Tison, Ed., vol. 2378. Springer, 2002, pp. 252–266.

[13] O. Bournez and F. Garnier, "Proving positive almost-sure termination," in *Proc. of RTA 2005*, ser. Lecture Notes in Computer Science, J. Giesl, Ed., vol. 3467. Springer, 2005, pp. 323–337.

[14] C. Faggian, "Probabilistic rewriting: Normalization, termination, and unique normal forms," in *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*, 2019, pp. 19:1–19:25.

[15] M. Avanzini, U. D. Lago, and A. Yamada, "On probabilistic term rewriting," *Sci. Comput. Program.*, vol. 185, 2020.

[16] C. Faggian and S. R. D. Rocca, "Lambda calculus and probabilistic computation," in *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, 2019, pp. 1–13.

[17] J. Koslowski, "A monadic approach to polycategories," *Theory and Applications of Categories*, vol. 14, no. 7, pp. 125–156, 2005.

[18] M. Weber, "Generic morphisms, parametric representations and weakly cartesian monads," *Theory Appl. Categ*, vol. 13, no. 14, pp. 191–234, 2004.

[19] M. M. Clementino, D. Hofmann, and G. Janelidze, "The monads of classical algebra are seldom weakly cartesian," *Journal of Homotopy and Related Structures*, vol. 9, no. 1, pp. 175–197, 2014.

[20] S. Szawiel and M. Zawadowski, "Monads of regular theories," *Applied Categorical Structures*, vol. 23, no. 3, pp. 215–262, 2015.

[21] N. D. Gautam, "The validity of equations of complex algebras," *Archiv für mathematische Logik und Grundlagenforschung*, vol. 3, no. 3, pp. 117–124, 1957.

[22] G. Grätzer and H. Lakser, "Identities for globals (complex algebras) of algebras," in *Colloquium Mathematicum*, vol. 56, no. 1, 1988, pp. 19–29.

[23] G. Plotkin, "Call-by-name, call-by-value and the lambda-calculus," *Theoretical Computer Science*, vol. 1, no. 2, pp. 125 – 159, 1975.

[24] M. Barr, "Relational algebras," *Lect. Notes Math.*, vol. 137, pp. 39–55, 1970.

[25] A. Kurz and J. Velebil, "Relation lifting, a survey," *J. Log. Algebr. Meth. Program.*, vol. 85, no. 4, pp. 475–499, 2016.

[26] D. Hofmann, G. Seal, and W. Tholen, Eds., *Monoidal Topology. A Categorical Approach to Order, Metric, and Topology*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2014, no. 153.

[27] D. Hoffman, "A cottage industry of lax extensions," *Categories and General Algebraic Structures with Applications*, vol. 3, no. 1, pp. 113–151, 2015.

[28] A. Thijs, *Simulation and fixpoint semantics*. Rijksuniversiteit Groningen, 1996.

[29] J. Hughes and B. Jacobs, "Simulations in coalgebra," *Theor. Comput. Sci.*, vol. 327, no. 1-2, pp. 71–108, 2004.

[30] J. Marti and Y. Venema, "Lax extensions of coalgebra functors and their logic," *J. Comput. Syst. Sci.*, vol. 81, no. 5, pp. 880–900, 2015.

[31] A. Baltag, "A logic for coalgebraic simulation," *Electr. Notes Theor. Comput. Sci.*, vol. 33, pp. 42–60, 2000.

[32] U. Dal Lago, F. Gavazzo, and P. Levy, "Effectful applicative bisimilarity: Monads, relators, and howe's method," in *Proc. of LICS 2017*, 2017, pp. 1–12.

[33] U. Dal Lago and F. Gavazzo, "Effectful normal form bisimulation," in preparation.

[34] J. Goubault-Larrecq, S. Lasota, and D. Nowak, "Logical relations for monadic types," *Mathematical Structures in Computer Science*, vol. 18, no. 6, pp. 1169–1217, 2008.

[35] R. C. Backhouse, P. J. de Bruin, P. F. Hoogendijk, G. Malcolm, E. Voermans, and J. van der Woude, "Polynomial relators (extended abstract)," in *Algebraic Methodology and Software Technology (AMAST '91), Proceedings of the Second International Conference on Methodology and Software Technology, Iowa City, USA, 22-25 May 1991*, ser. Workshops in Computing, M. Nivat, C. Rattray, T. Rus, and G. Scollo, Eds. Springer, 1991, pp. 303–326.

[36] R. C. Backhouse and P. F. Hoogendijk, "Elements of a relational theory of datatypes," in *Formal Program Development - IFIP TC2/WG 2.1 State-of-the-Art Report*, ser. Lecture Notes in Computer Science, B. Möller, H. Partsch, and S. A. Schuman, Eds., vol. 755. Springer, 1993, pp. 7–42.

[37] R. S. Bird and O. de Moor, *Algebra of programming*, ser. Prentice Hall International series in computer science. Prentice Hall, 1997.

[38] J. Beck, "Distributive laws," in *Seminar on triples and categorical homology theory*. Springer, 1969, pp. 119–140.

[39] D. Varacca and G. Winskel, "Distributing probability over non-determinism," *Math. Struct. Comput. Sci.*, vol. 16, no. 1, pp. 87–113, 2006.

[40] D. Varacca, "Probability, nondeterminism and concurrency: two denotational models for probabilistic computation," Ph.D. dissertation, Aarhus University, 2003.

[41] B. Klin and J. Salamanca, "Iterated covariant powerset is not a monad." in *MFPS*, 2018, pp. 261–276.

[42] H. P. Sankappanavar and S. Burris, "A course in universal algebra," *Graduate Texts Math*, vol. 78, 1981.

[43] T. Leinster, *Higher Operads, Higher Categories*, ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

[44] B. Davey and H. Priestley, *Introduction to lattices and order*. Cambridge University Press, 1990.

[45] A. Syropoulos, "Mathematics of multisets," in *Workshop on Membrane Computing*. Springer, 2000, pp. 347–358.

[46] M. H. Stone, "Postulates for the barycentric calculus," *Annali di Matematica Pura ed Applicata*, vol. 29, no. 1, pp. 25–30, 1949.

[47] G. Schmidt, *Relational Mathematics*, ser. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011, vol. 132.

[48] M. Bezem, J. Klop, E. Barendsen, R. de Vrijer, and Terese, *Term Rewriting Systems*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003. [Online]. Available: https://books.google.it/books?id=7QQ5u-4tRUkC

[49] B. Jacobs, "Trace semantics for coalgebras," in *Proceedings of the Workshop on Coalgebraic Methods in Computer Science, CMCS 2004, Barcelona, Spain, March 27-29, 2004*, 2004, pp. 167–184.

[50] L. Parlant, J. Rot, A. Silva, and B. Westerbaan, "Preservation of equations by monoidal monads," in *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, 2020, pp. 77:1–77:14.

[51] U. Dal Lago and M. Zorzi, "Probabilistic operational semantics for the lambda calculus," *RAIRO - Theor. Inf. and Applic.*, vol. 46, no. 3, pp. 413–450, 2012.

[52] B. Accattoli, C. Faggian, and G. Guerrieri, "Factorize factorization," in *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, ser. LIPIcs, vol. 183. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 6:1–6:25. [Online]. Available: https://doi.org/10.4230/LIPIcs.CSL.2021.6

[53] S. R. D. Rocca and L. Paolini, *The Parametric Lambda Calculus - A Metamodel for Computation*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[54] C. Lüth, "Categorical term rewriting : monads and modularity," Ph.D. dissertation, University of Edinburgh, UK, 1998.

[55] C. Lüth and N. Ghani, "Monads and modular term rewriting," in *Category Theory and Computer Science, 7th International Conference, CTCS '97, Santa Margherita Ligure, Italy, September 4-6, 1997, Proceedings*, 1997, pp. 69–86.

[56] T. Fritz and P. Perrone, "Monads, partial evaluations, and rewriting," *Electronic Notes in Theoretical Computer Science*, vol. 352, pp. 129 – 148, 2020, the 36th Mathematical Foundations of Programming Semantics Conference, 2020.

[57] U. De Liguoro and A. Piperno, "Non deterministic extensions of untyped lambda-calculus," *Inf. Comput.*, vol. 122, no. 2, pp. 149–177, 1995.

[58] P. Johann, A. Simpson, and J. Voigtländer, "A generic operational metatheory for algebraic effects," in *Proc. of LICS 2010*. IEEE Computer Society, 2010, pp. 209–218.

[59] A. Lopez and A. Simpson, "Basic operational preorders for algebraic effects in general, and for combined probability and nondeterminism in particular," in *Proc. of CSL 2018*, 2018, pp. 29:1–29:17.

Appendix
Proofs of Section IV

Sometimes, it will be useful to work with the equivalent description of monads is as Kleisli triples [8]. A *Kleisli triple* $\mathbb{T} = (T, \eta, -^\dagger)$ consists of a map T on sets, a family of functions $\eta : X \to TX$, and an operation mapping each function $f : X \to TY$ to its Kleisli extension $f^\dagger : TX \to TY$ subject to the following laws: $\eta^\dagger = 1_T$; $f^\dagger \circ \eta = f$; $(g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$. In particular, given a monad $\mathbb{T} = (T, \eta, \mu)$, notice that $f^\dagger$ is defined as $\mu \circ Tf$. Accordingly, algebraic operations can be equivalently described as families of maps $f_X : (TX)^n \to TX$ such that $g^\dagger \circ f_X = f_Y \circ \prod_n g^\dagger$.

$$\begin{tikzcd}
(TX)^n \arrow[r, "f"] \arrow[d, "\prod_n g^\dagger"'] & TX \arrow[d, "g"] \\
(TY)^n \arrow[r, "f"'] & TY
\end{tikzcd}$$

Given a monad $\mathbb{T} = (T, \eta, \mu)$, let Γ be a relational extension of T. Recall that Γ is a relational extension of $\mathbb{T}$ if the following laws hold:

$$R; \eta = \eta; \Gamma R \qquad\qquad \Gamma\Gamma R; \mu = \mu; \Gamma R.$$

Diagrammatically:

$$\begin{tikzcd}
X \arrow[r, "\eta"] \arrow[d, "R"'] & TX \arrow[d, "\Gamma R"] \\
Y \arrow[r, "\eta"'] & TY
\end{tikzcd}
\qquad
\begin{tikzcd}
TTX \arrow[r, "\mu"] \arrow[d, "\Gamma\Gamma R"'] & TX \arrow[d, "\Gamma R"] \\
TTY \arrow[r, "\mu"'] & TY
\end{tikzcd}$$

Lemma 44. *Let Γ be a relational extension of $\mathbb{T} = (T, \eta, \mu)$. We have:*

$$R; g \subseteq f; \Gamma S \implies \Gamma R; g^\dagger \subseteq f^\dagger; \Gamma S.$$

Diagrammatically:

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "R"'] \arrow[dr, phantom, "\subseteq"] & TZ \arrow[d, "\Gamma S"] \\
Y \arrow[r, "g"'] & TW
\end{tikzcd}
\implies
\begin{tikzcd}
TX \arrow[r, "f^\dagger"] \arrow[d, "\Gamma R"'] \arrow[dr, phantom, "\subseteq"] & TZ \arrow[d, "\Gamma S"] \\
TY \arrow[r, "g^\dagger"'] & TW
\end{tikzcd}$$

Proof. We have:

$$\begin{aligned}
\Gamma R; g^\dagger &= \Gamma R; Tg; \mu \\
&= \Gamma(R; g); \mu \\
&\subseteq \Gamma(f; \Gamma S); \mu \\
&= Tf; \Gamma\Gamma S; \mu \\
&= Tf; \mu; \Gamma S \\
&= f^\dagger; \Gamma S
\end{aligned}$$

□

Lemma 45. *Let Γ be a relational extension of $\mathbb{T}$. Then, for any $R : X \nrightarrow Y$ and any algebraic operation $f : T^n \Rightarrow T$, we have: $\prod_i \Gamma R; f \subseteq f; \Gamma R$.*

Proof. The proof uses the correspondence between algebraic operations and generic effects [9]. We use such a correspondence implicitly, this way giving a self-contained proof. We have to show that for all $x : \boldsymbol{n} \to TX$, $y : \boldsymbol{n} \to TY$, if $x(i)\Gamma R y(i)$, for any $i \in \boldsymbol{n}$, then $f(x)\Gamma R f(y)$. By Lemma 44, we have:

$$\begin{tikzcd} \boldsymbol{n} \arrow[r, "x"] \arrow[d, "I"'] \arrow[dr, phantom, "\subseteq"] & TX \arrow[d, "\Gamma R"] \\ \boldsymbol{n} \arrow[r, "y"'] & TY \end{tikzcd} \implies \begin{tikzcd} T\boldsymbol{n} \arrow[r, "x^\dagger"] \arrow[d, "I"'] \arrow[dr, phantom, "\subseteq"] & TX \arrow[d, "\Gamma R"] \\ T\boldsymbol{n} \arrow[r, "y^\dagger"'] & TY \end{tikzcd}$$

Since $\forall i \in \boldsymbol{n}.\ x(i)\ \Gamma R\ y(i)$ means nothing but $I; y \subseteq x; \Gamma R$, we infer the consequent of the above implication. Let us now consider $\eta : \boldsymbol{n} \to T\boldsymbol{n}$, so that $f(\eta) \in T\boldsymbol{n}$. Then, $I; y^\dagger \subseteq x^\dagger; \Gamma R$ implies $x^\dagger(f(\eta))\ \Gamma R\ y^\dagger(f(\eta))$. Since f is algebraic, $x^\dagger(f(\eta)) = f(x^\dagger \circ \eta) = f(x)$ (and similarly for y). □

Proposition 14. *Let $\mathbb{T}$ be a monad with relational extension Γ. Let $\Delta = \Gamma(-); \mu$. Then, for any $R : X \nrightarrow Y$ and algebraic operation $f : T^n \Rightarrow T$, we have:*

$$\prod_i \Delta R; f \subseteq f; \Delta R \qquad\qquad \eta; \Delta R = R.$$

Diagrammatically:

$$\begin{tikzcd} (TX)^n \arrow[r, "f"] \arrow[d, "\prod_{i=1}^n \Delta R"'] \arrow[dr, phantom, "\subseteq"] & TX \arrow[d, "\Delta R"] \\ (TY)^n \arrow[r, "f"'] & TY \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "\eta"] \arrow[dr, "R"'] & TX \arrow[d, "\Delta R"] \\ & TY \end{tikzcd}$$

Proof. For the second equality, we have: $\eta; \Delta R = \eta; \Gamma R; \mu = R; \eta; \mu = R$. For the first one, we chase the following diagram relying on algebraicity of f and Lemma 45.

$$\begin{tikzcd} (TX)^n \arrow[r, "f"] \arrow[d, "\prod_i \Gamma R"'] & TX \arrow[d, "\Gamma R"] \\ (TTY)^n \arrow[r, "f"'] \arrow[d, "\prod_i \mu_i"'] & (TTY)^n \arrow[d, "\prod_i \mu_i"] \\ (TY)^n \arrow[r, "f"'] & TY \end{tikzcd}$$

□

Proposition 15. *Let Γ be a relational extension of a* functor *T which is part of a monad $\mathbb{T} = (T, \eta, \mu)$. Then, Γ is a relational extension of $\mathbb{T}$ iff the following laws hold, for all $R : X \nrightarrow Y$, $S : Y \nrightarrow Z$, and $P : Z \nrightarrow W$:*

$$R \cdot \eta_Y = R \quad \eta_X \cdot S = S \quad R \cdot (S \cdot P) = (R \cdot S) \cdot P.$$

Proof. We have to show that the equalities in the statement of the theorem hold iff we have $Q; \eta = \eta; \Gamma Q$ and $\Gamma\Gamma Q; \mu = \mu; \Gamma Q$, (for all $Q : X \nrightarrow Y$). First, we notice that $R \cdot \eta_Y = R$ always holds. Next, we show that the law $\eta_X \cdot S = S$ is equivalent to $Q; \eta = \eta; \Gamma Q$. Assuming the latter, we have

$$\eta \cdot S = \eta; \Gamma S; \mu = S; \eta; \mu = S.$$

Assuming the former, we have:

$$Q; \eta = \eta \cdot (Q; \eta) = \eta; \Gamma Q; T\eta; \mu = \eta; \Gamma Q.$$

Finally, we show that the law $R \cdot (S \cdot P) = (R \cdot S) \cdot P$ is equivalent to $\Gamma\Gamma Q; \mu = \mu; \Gamma Q$. Assuming the former, we obtain the desired equality by taking $R = I_{TTX}$, $S = I_{TX}$, $P = Q; \eta$. Assuming the latter, we have:

$$\begin{aligned} R \cdot (S \cdot P) &= R; \Gamma(S; \Gamma P; \mu); \mu \\ &= R; \Gamma S; \Gamma\Gamma P; T\mu; \mu \\ &= R; \Gamma S; \Gamma\Gamma P; \mu; \mu \\ &= R; \Gamma S; \mu; \Gamma P; \mu \\ &= (R \cdot S) \cdot P. \end{aligned}$$

□

Proposition 21. *Given $R : X \nrightarrow Y$, $S : Y \nrightarrow Z$, and $P : X \nrightarrow Y$, $Q : Y \nrightarrow Z$, we have:*

$$R; \widetilde{T}S \subseteq P; \widetilde{T}Q \implies \widetilde{T}R; \widetilde{T}S \subseteq \widetilde{T}P; \widetilde{T}Q.$$

Proof. We have:

$$\begin{aligned} \widetilde{T}R; \widetilde{T}S &= \hat{T}R; \mu; \hat{T}S \\ &= \hat{T}R; \hat{T}\hat{T}S; \mu; \mu \\ &= \hat{T}R; \hat{T}\hat{T}S; T\mu; \mu \\ &= \hat{T}(R; \hat{T}S; \mu); \mu \\ &= \hat{T}(R; \widetilde{T}S); \mu \\ &\subseteq \hat{T}(P; \widetilde{T}Q); \mu \\ &= \hat{T}(P; \hat{T}Q; \mu); \mu \\ &= \hat{T}P; \hat{T}\hat{T}Q; T\mu; \mu \\ &= \hat{T}P; \hat{T}\hat{T}Q; \mu; \mu \\ &= \hat{T}P; \mu; \hat{T}Q; \mu \\ &= \widetilde{T}P; \widetilde{T}Q. \end{aligned}$$

□

Theorem 22. *The pointed diamond technique*

$$R^-; S \subseteq \widetilde{T}S; (\widetilde{T}R)^- \implies (\widetilde{T}R)^-; \widetilde{T}S \subseteq \widetilde{T}S; (\widetilde{T}R)^-$$

is sound.

Proof. Assume $R^-;S \subseteq \widetilde{T}S;(\widetilde{T}R)^-$. By standard calculations based on the properties of $\hat{T}$ and the hypothesis $R^-;S \subseteq \widetilde{T}S;(\widetilde{T}R)^-$, we have:

$$\begin{aligned}(\widetilde{T}R)^-;\widetilde{T}S &= (\hat{T}R;\mu)^-;\hat{T}S;\mu\\ &= \mu^-;\hat{T}(R^-);\hat{T}S;\mu\\ &= \mu^-;\hat{T}(R^-;S);\mu\\ &\subseteq \mu^-;\hat{T}(\widetilde{T}S;(\widetilde{T}R)^-);\mu\\ &= \mu^-;\hat{T}(\hat{T}S;\mu;(\hat{T}R;\mu)^-);\mu\\ &= \mu^-;\hat{T}\hat{T}S;\hat{T}\mu;\hat{T}\mu^-;\hat{T}\hat{T}R^-;\mu\\ &= \mu^-;\hat{T}\hat{T}S;T\mu;(T\mu)^-;\hat{T}\hat{T}R^-;\mu.\end{aligned}$$

Next, we notice that we have:

$$\mu^-;\hat{T}\hat{T}S \subseteq \hat{T}S;\mu^- \qquad \hat{T}\hat{T}R^-;\mu \subseteq \mu;\hat{T}R^-.$$

For instance, by shunting we have $\mu^-;\hat{T}\hat{T}S \subseteq \hat{T}S;\mu^-$ iff $\hat{T}\hat{T}S \subseteq \mu;\hat{T}S;\mu^-$. Since $\hat{T}\hat{T}S;\mu = \mu;\hat{T}S$, to prove $\hat{T}\hat{T}S \subseteq \mu;\hat{T}S;\mu^-$ it is enough to show $\hat{T}\hat{T}S \subseteq \hat{T}\hat{T}S;\mu;\mu^-$, which indeed holds since $I \subseteq \mu;\mu^-$. In a similar way, one can prove $\hat{T}\hat{T}R^-;\mu \subseteq \mu;\hat{T}R^-$. Using these inclusions, we thus obtain:

$$\mu^-;\hat{T}\hat{T}S;T\mu;(T\mu)^-;\hat{T}\hat{T}R^-;\mu \subseteq \hat{T}S;\mu^-;T\mu;(T\mu)^-;\mu;\hat{T}R^-.$$

Now we focus on the relation $\mu^-;T\mu;(T\mu)^-;\mu$. Diagrammatically:

$$\begin{tikzcd} TTA \arrow[r, "(\mu_{TA})^-", "\shortmid"{marking}] & TTTA \arrow[d, "T\mu_A"] \\ TTTA \arrow[u, "\mu_{TA}"] & TTA \arrow[l, "T\mu_A"', "\shortmid"{marking}] \end{tikzcd}$$

Consider the maps $TTA \xrightarrow{\mu_A} TA \xleftarrow{\mu_A} TTA$ and $TTA \xleftarrow{\mu_{TA}} TTTA \xrightarrow{T\mu_A} TTA$ and the naturality square $T\mu_A;\mu_A = \mu_{TA};\mu_A$. Since $\mathbb{T}$ is weakly cartesian (and thus μ is), we infer

$$(T\mu_A)^-;\mu_{TA} = \mu_A;(\mu_A)^- \quad (\mu_{TA})^-;T\mu_A = \mu_A;(\mu_A)^-.$$

Applying these identities (as well as $\mu^-;\mu \subseteq I$), we obtain

$$\mu^-;T\mu;(T\mu)^-;\mu = \mu;\mu^-;\mu;\mu^- \subseteq \mu;\mu^-.$$

We can thus complete our calculation of $(\widetilde{T}R)^-;\widetilde{T}S$ by inferring

$$\begin{aligned}(\widetilde{T}R)^-;\widetilde{T}S \subseteq \cdots &\subseteq \hat{T}S;\mu^-;T\mu;(T\mu)^-;\mu;\hat{T}R^-\\ &\subseteq \hat{T}S;\mu;\mu^-;\hat{T}R^-\\ &= \hat{T}S;\mu;(\hat{T}R;\mu)^-\\ &= \widetilde{T}S;(\widetilde{T}R)^-\end{aligned}$$

□

Theorem 25. *Let $\mathbb{T} = (\Sigma, E)$ be a theory. If the power law is a distributive law, then for any $\mathbb{F}_{\mathbb{T}}$-corelation $R : X \nrightarrow Y$ we have:*

$$[t]\ \widetilde{F_{\mathbb{T}}}R\ [s] \iff t\ R^{\Sigma}\ [s].$$

Proof. Let δ be the power law $\hat{F}_{\mathbb{T}}(\ni)$. If δ is a distributive law, then $\mathcal{P}^{\delta}$ extends from Σ-**Alg** to (Σ, E)-**Alg**. In fact, δ being distributive law, it gives a lifting of $\mathbb{P}$ to $\mathbf{EM}(F_{\mathbb{T}}) \cong (\Sigma, E)$-**Alg**. δ being the power law, such a lifting is nothing but $\mathcal{P}^{\delta}$. In particular, $\mathcal{P}F_{\mathbb{T}}Y$ carries a (Σ, E)-algebra structure, so that we we can exploit the free algebra properties of $T_{\Sigma}X$ and $F_{\mathbb{T}}X$, obtaining:

$$\begin{tikzcd} X \arrow[r, hook] \arrow[dr, "R"'] & T_{\Sigma}X \arrow[r, "{[-]}"] \arrow[d, dotted, "R^{\Sigma}"] & F_{\mathbb{T}}X \arrow[dl, dotted, "\widetilde{F_{\mathbb{T}}}R"] \\ & \mathcal{P}F_{\mathbb{T}}X & \end{tikzcd}$$

□

Remark 46 (Affine equations)**.** Example 28 shows that most of the usual equational theories used to model computational effects are either linear or can be made linear, at a reasonable price. A notable exception to this pattern is the theory of the reader monad (as well as the theory of the global state monad) which is made of non-linear equations only, namely: $\mathtt{rd}(x,x) = x$ and $\mathtt{rd}(\mathtt{rd}(x,w),\mathtt{rd}(z,y)) = \mathtt{rd}(x,y)$. However, for the kind of examples we have seen so far, only the first equation (namely idempotency of $\mathtt{rd}$) is problematic. This suggests that problems are not given by non-linear equations, but by non-*affine* equations. In an affine equation, the variables in the two terms must be distinct, but each of these variables must appear *at most* once in each term. Theorem 26 does not hold for affine theories, however it does hold if we replace the powerset monad with the *non-empty* powerset monad $\mathbb{P}^+$ [50]. As a consequence, we obtain an extension of Corollary 27 to affine equations, provided that we restrict to $\mathbb{F}_{\mathbb{T}}$-corelations $R : X \nrightarrow Y$ such that $R[x] \neq \emptyset$, for any set x. Since we will endow λ-calculi with $\mathbb{F}_{\mathbb{T}}$-corelations $R : \Lambda \nrightarrow \Lambda$ satisfying such a property, the results in Section V which we are going to introduce can be easily extended to *affine* equational theories.

Appendix
Proofs of Section V

Lemma 47. $\overset{\neg\mathtt{s}}{\Rightarrow}_{\beta};\Rightarrow_{\Sigma} \subseteq \Rightarrow_{\Sigma};\Rightarrow_{\beta}$.

Proof. First, we observe that $\overset{\neg\mathtt{s}}{\to}_{\beta};\Rightarrow_{\sigma} \subseteq \to_{\sigma};\Rightarrow_{\beta}$, which is immediate to check since $\overset{\neg\mathtt{s}}{\to}_{\beta}$ preserves the shape of terms. We then conclude the desired result by Proposition 21. □

Theorem 38 (Surface Standardization)**.** $\Rightarrow^* \subseteq \overset{\mathtt{s}}{\Rightarrow}^*;\overset{\neg\mathtt{s}}{\Rightarrow}^*$.

Proof. From Lemma 37 we infer $\Rightarrow^*_{\beta} \subseteq \overset{\mathtt{s}}{\Rightarrow}^*_{\beta};\overset{\neg\mathtt{s}}{\Rightarrow}^*_{\beta}$. We conclude the thesis from Lemma 47, using Lemma 34. □

Lemma 41. *Let $\gamma, \delta \in \Sigma \cup \{\beta\}$, $\rightarrowtail = \widetilde{F_{\mathbb{T}}}(\to)$, and $\mathtt{t}$ be a term with two distinct* surface *redexes. If $[\theta(\mathtt{t}_1,\ldots,\mathtt{t}_n)]\ {}_{\gamma}\overset{\mathtt{s}}{\leftarrow}\ \mathtt{t}\ \overset{\mathtt{s}}{\to}_{\delta}\ [\vartheta(\mathtt{s}_1,\ldots,\mathtt{s}_m)]$ and θ commutes with ϑ, then $[\theta(\mathtt{t}_1,\ldots,\mathtt{t}_n)] \overset{\mathtt{s}}{\rightarrowtail}_{\delta} [p]\ {}_{\gamma}\overset{\mathtt{s}}{\leftarrowtail} [\vartheta(\mathtt{s}_1,\ldots,\mathtt{s}_m)]$.*

Proof. The proof is by induction on $\mathtt{t}$. First, observe that since reductions are surface, $\mathtt{t}$ cannot be neither a variable nor a λ-abstraction. Additionally, $\mathtt{t}$ cannot be itself a redex, as neither $(\lambda\mathtt{x}.\mathtt{s})\mathtt{v}$ nor $\sigma(\mathtt{t}_1,\ldots,\mathtt{t}_n)$ can contain surface redexes. Therefore, $\mathtt{t}$ must be of the form $\mathtt{pq}$. We now proceed by case analysis. If both redexes are inside $\mathtt{p}$ or $\mathtt{q}$, then we are done

by induction hypothesis. Otherwise, one redex is inside $\mathtt{p}$ and the other is inside $\mathtt{q}$. Without loss of generality we can assume $\mathtt{p} = \mathtt{S}_1\langle\mathtt{s}\rangle$ (and $\mathtt{s}$ γ-redex) and $\mathtt{q} = \mathtt{S}_2\langle\mathtt{r}\rangle$ (and $\mathtt{r}$ δ-redex). Therefore, we have:

$$\begin{aligned}\mathtt{t} &\xrightarrow{\mathtt{s}}_\gamma [\theta(\ldots, \mathtt{S}_1\langle\mathtt{s}_i\rangle\mathtt{q}, \ldots)]\\ &\overset{\mathtt{s}}{\rightarrowtail}_\delta [\theta(\ldots, [\vartheta(\ldots, \mathtt{S}_1\langle\mathtt{s}_i\rangle\mathtt{S}_2\langle\mathtt{r}_j\rangle, \ldots)], \ldots)]\end{aligned}$$

Similarly, we have

$$\begin{aligned}\mathtt{t} &\xrightarrow{\mathtt{s}}_\delta [\vartheta(\ldots, \mathtt{p}\mathtt{S}_2\langle\mathtt{r}_j\rangle, \ldots)]\\ &\overset{\mathtt{s}}{\rightarrowtail}_\gamma [\vartheta(\ldots, [\theta(\ldots, \mathtt{S}_1\langle\mathtt{s}_i\rangle\mathtt{S}_2\langle\mathtt{r}_j\rangle, \ldots)] \ldots)]\end{aligned}$$

We conclude the thesis since θ commutes with ϑ. □

Proposition 42 (Surface is $\diamond$)**.** *If $\mathbb{T} = (\Sigma, E)$ is commutative, then then $\overset{\mathtt{s}}{\Rightarrow}$ is diamond, and thus confluent. The same holds, in particular, for $\Rightarrow_\Sigma$.*

Proof. From Lemma 41 we infer $(\overset{\mathtt{s}}{\rightarrow}\cup\eta)^-; (\overset{\mathtt{s}}{\rightarrow}\cup\eta) \subseteq \widetilde{F_{\mathbb{T}}}(\overset{\mathtt{s}}{\rightarrow}\cup\eta); \widetilde{F_{\mathbb{T}}}(\overset{\mathtt{s}}{\rightarrow}\cup\eta)^- = \overset{\mathtt{s}}{\Rightarrow}; \overset{\mathtt{s}}{\Leftarrow}$. By Theorem 22 we thus conclude $\overset{\mathtt{s}}{\Leftarrow}; \overset{\mathtt{s}}{\Rightarrow} \subseteq \overset{\mathtt{s}}{\Rightarrow}; \overset{\mathtt{s}}{\Leftarrow}$. Finally, notice that $\Rightarrow_\Sigma = \overset{\mathtt{s}}{\Rightarrow}_\Sigma$. □

We now prove Theorem 43.

Theorem 43 (Confluence)**.** *If $\mathbb{T} = (\Sigma, E)$ is commutative, then $\Rightarrow$ is confluent.*

Our proof relies on the Hindley-Rosen Lemma.

Lemma 48 (Hindley-Rosen)**.** *Let A be a set, and $R, S : A \nrightarrow A$ be binary relations on A. Then, $R \cup S$ is confluent if both R and S are confluent, and R and S commute.*

Since $\Rightarrow\, = \,\Rightarrow_\beta \cup \Rightarrow_\Sigma$, by Lemma 48, to prove confluence it is enough to show that $\Rightarrow_\beta$ and $\Rightarrow_\Sigma$ are confluent and commute. For the commutation of $\Rightarrow_\beta$ and $\Rightarrow_\Sigma$ we rely on Theorem 22

Lemma 49. *The relations $\Rightarrow_\beta$ and $\Rightarrow_\Sigma$ $\diamond$-commute.*

Proof. By Theorem 22, it is enough to prove that $(\rightarrow_\beta \cup \eta)^-; (\rightarrow_\Sigma\cup\eta) \subseteq \Rightarrow_\Sigma; (\Rightarrow_\beta)^-$. We proceed by cases on $\rightarrow_\beta\cup\eta$ and $\rightarrow_\Sigma \cup\, \eta$, the only non trivial one being $\rightarrow_\beta{}^-; \rightarrow_\Sigma \subseteq \Rightarrow_\Sigma; (\Rightarrow_\beta)^-$. The latter amounts to show that if $s_1 \;{}_\beta\!\leftarrow \mathtt{t} \rightarrow_\Sigma s_2$, then $s_1 \Rightarrow_\Sigma p \;{}_\beta\!\Leftarrow s_2$. By Lemma 41 we have to check only cases for deep reductions. Since Σ-reductions are surface, $\mathtt{t}$ must be of the form $\sigma(\mathtt{s}_1, \ldots, \mathtt{s}_n)$ with a $\mathtt{s}_i$ containing the β-redex. We are done since σ-reductions always commute with β-ones. □

Next, we show that $\Rightarrow_\beta$ is confluent.

Proposition 50 ($\Rightarrow_\beta$-confluence)**.** *The relation $\Rightarrow_\beta$ is confluent.*

Proof sketch. From confluence of β-reduction in the pure call-by-value λ-calculus, using Lemma 37. □

We can finally prove Theorem 43 using the Hindley-Rosen Lemma, together with Lemma 49, Proposition 50, and Proposition 42.