Kleene monads

Sergey Goncharov

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften
– Dr.-ing. –

Vorgelegt im Fachbereich 3 (Mathematik und Informatik)
der Universität Bremen
im April 2010

"I think we agree, the past is over."

George W. Bush, *On his meeting with John McCain,*
May 10, 2000

Abstract

The thesis is devoted to the problem of reasoning about computations with side-effects, i.e. computations featuring memory states, non-determinism, non-termination, etc. The notion of a side-effect is known to be well formally captured by the concept of the computational monad. This gives rise to the 'computations as monads' paradigm, which is adopted here. Several classes of computational monads, such as (strong) additive monads and (strong) Kleene monads, were introduced and studied, with a special focus on the problem of logical tractability. The developed formal device for proving equational properties of monad classes combines rewriting techniques with inductive reasoning. This approach turned out to be successful and resulted in a number of decidability and undecidability theorems.

Contents

Chapter 0

Introduction

From the very beginning of the history of systematic scientific study of the process of computation, researchers were not so much concerned about the structural side of the issue. This was partly because much of the theory was developed before any practical implementations, and researchers had not yet faced the problems related to the use of programming in industry. The theory of computations was originally developed as a pure mathematical discipline whose terms and methods were chosen in favour of simplification of the theoretical treatment. Another reason why the structural side of programming was originally ignored was because the depth of the problems about computations were not immediately realised, e.g. if Hilbert's program turned out to be successful, one would not need to care so much about programming language implementations.

Since the problems concerning programming generally turned out to be very hard, and efforts to reduce them to some simple common basis produced no positive results, much effort was invested in broadening the notion of computation rather than narrowing it. A perfect example of such broadening was Domain Theory, developed by Dana Scott, which resulted in the Scott Thesis as a counterpart to the Turing Thesis. In contrast to the Turing Thesis, which states that computable functions are those and only those which are recursive, that Scott Thesis states that computable functions are precisely those which are continuous in the sense of Domain Theory. Whereas reasoning about programs in terms of computability theory always implies an encoding of the programs under consideration into the language of recursive functions (or some equivalent formalism), with no direct interpretation relevant for practical programming, Domain Theory only requires keeping track of continuity, a notion that already makes perfect sense for program code written in conventional programming languages. Moreover, Domain Theory provides a large framework of abstract constructions reflecting the process of program composition and usage. All this makes Domain Theory a very flexible tool.

Another important landmark in formalising the notion of computation was the introduction of categorical semantics for programs, instead of set-theoretic semantics. Categorical semantics turned out to be exceptionally advantageous and successful. This happened chiefly because, according to the categorical treatment of programs, the difference between the data and the function becomes explicit on a very basic level. In addition, category theory provides on the core level a suitable idea of modularity: it allows reusing the same abstract constructions over and over regardless of the precise settings.

Only by using the language of category theory could the notion of computational effect be uniformly formalised, which had constantly slipped away when either the set-theoretic or order-theoretic approach was used. This formalisation was originated by Moggi in his seminal paper [Mog91]. Moggi matched the formal notion of the monad from category theory with the informal notion of a side-effect from programming practice. This correspondence is of course by no means a provable fact but rather a matter of beliefs, though commonly accepted. For this reason, the claim that 'a computational effect is a monad' is sometimes referred to as Moggi's Principle (e.g. [Fil94]).

Moggi's original proposal was to use the notion of the computational monad to justify the denotational semantics of functional programming languages with side-effects like ML. Sure enough, the actual usage of this notion went beyond these bounds rather quickly. We recall several principal developments of this work. After Moggi's seminal work, computational monads were quickly popularised by Wadler and others [Wad92, Wad94, Wad95] as a technique for structuring functional programs. This resulted in the explicit introduction of monads in the pure functional programming language Haskell, where they finally settled as one of the most important features [PJW93]. A remarkable application of computational monads in denotational semantics was to model the ambiguity of evaluation order rather than a specific computational effect [PM00]. In particular, this made it possible to give a formal semantics of the C programming language [Pap98]. The monadic view turned out to be helpful in programming language design, e.g. it is used to simplify the type-systems for region-based languages like ML-Kit and Cyclone [FM06, FMA06]. A separate line of research in computational monads involves monad-based program logics, which are intended to empower classical program logics such as Hoare logic, dynamic logic, etc., with side-effecting computations [Pit91, Mog95, SM04].

This thesis is most closely concerned with the latter line of research. It turned out that introducing a recursion operator into the settings of monad-based program logics made the analysis of the metalogical properties of the resulting system highly non-trivial. In particular, the questions of soundness, completeness and logical tractability raised substantial difficulties. Therefore, it appeared reasonable to isolate a small part of monad-based program logics, yet expressive enough to support the program structure of interest, and study it individually. Generally, the equational part is formulated in terms of equational logic, whereas the more advanced program logics that can incorporate it may feature modal operators, etc. The relationship between the equational part of a program logic and the program logic itself is thus of the same kind as, e.g., the relationship between Kleene algebra and propositional dynamic logic.

In order to implement the outlined approach we introduce a notion of a strong Kleene monad as a generic monad supporting iterative execution of programs. We introduce a more specific notion of an ω-continuous strong Kleene monad, which has an order-theoretic implementation of the iteration operator. We also distinguish a notion of a strong additive monad, which implements finitary nondeterminism but not iteration. The latter notion is used as an auxiliary device for studying Kleene monads but is nonetheless of interest in itself. The summary of main results achieved in the thesis is as follows.

– A strongly normalising rewriting system capturing the equational theory of strong monads over cartesian categories (Theorem 1.24).

– A strongly normalising rewriting system capturing the equational theory of strong additive monads over cartesian categories (Theorem 2.17).

– Non-recursive enumerability of the the equational theory of strong ω-continuous Kleene monads (Theorem 3.42).

– A completeness theorem for a fragment of an equational theory of strong continuous Kleene monads (Theorem 4.49).

Acknowledgements

Support by the Universität Bremen under ZF Grant Nr. 03/109/04 (”Monadic computational logic in HOL”, Lutz Schröder) and a follow-up PhD scholarship, and by the Deutsche Forschungsgemeinschaft under Grant Nr. LU 707/9-1 (”A monad-based program logic construction kit for generic and heterogeneous side effects”, Christoph Lüth and Lutz Schröder) is gratefully acknowledged.

The author also would like to thank personally Till Mossakowski, Christoph Lüth and Maxim Bortin for valuable discussions and express special gratitude to Lutz Schröder, the scientific supervisor of the work whose empathy seems bottomless.

Chapter 1

Monads for computations

1.1 General qualifications and conventions

The notation used in this thesis, especially in those parts of it which are concerned with equational logic, is greatly influenced by [Cro94]. Treatment of internal languages of various monad classes follows the lines drawn by Moggi, [Mog89b, Mog91] with the adjustment of using Haskell's do-notation [SM03] instead of the original let-notation.

This thesis uses the following procedure for symbols and font faces: lower-case Roman letters stand for terms and programs; capital Roman letters stand for functors; bold capital Roman letters stand for categories, in particular, the category of sets shall be called **Set**; lower-case Greek letters stand for variable substitutions as well as for natural transformations; capital Greek letters are used for variable contexts; blackboard bold characters (like $\mathbb{T}$, $\mathbb{R}$, etc.) are reserved for monads.

Unless otherwise stated, we count the naturals starting from 0. The notation $\bar{t}$ shall denote the vector $(t_1, \ldots, t_n)$ where n occasionally will not be specified if it is implied by the context or is irrelevant. Given some vector $\bar{t}$ and an appropriate index i, we denote by $\bar{t}_{\hat{\imath}}$ the vector obtained from $\bar{t}$ by dropping the i-th component. We denote by $|\bar{t}|$ the length of a vector t and by $|S|$ the cardinality of a set S. Every statement involving free variables is supposed to be universally quantified over these variables, ranging over the values for which the statement makes sense. For example, 'in any monad $f^\dagger \circ \eta$ is equal to f' should be read 'in any monad $f^\dagger \circ \eta$ is equal to f for any morphism $f : A \to TB$ where A and B are some objects of the underlying category'.

The terms 'operator' as well as 'function' will be commonly used as synonyms for 'morphism', especially when the categorical interpretation coincides with the set-theoretic one. Similarly, we do not make any formal distinction between 'reduction' and 'rewriting', e.g. we use the terms 'reduction rule' and 'rewrite rule' interchangeably.

We will commonly omit subscripts referring to types, i.e. we write $\square$, τ, $\oplus$ instead of $\square_{A,B}$, $\tau_{A,B}$ and $\oplus_{A,B}$. Similarly, we omit superscripts referring to monad names, i.e. we write η, μ instead of $\eta^{\mathbb{T}}, \mu^{\mathbb{T}}$ if it does not cause ambiguity. We use '$\doteq$' as an abbreviation for 'syntactically equal' and '$:=$' as an abbreviation for 'equal by definition'. In particular, once we put $s := t$ at some point, $s \doteq t$ is supposed to hold henceforth (unless s is redefined).

There are few conventions concerning the usage of *variable substitutions*. A variable substitution, or simply a *substitution* is a map σ from some set of variables, called the *domain of* σ, to terms. We denote the result of applying a substitution σ to a term t by $t\sigma$. If the domain of a substitution σ is finite, e.g. $\{v_1, \dots, v_n\}$ we shall use the notation $[v_1\sigma/v_1, \dots, v_n\sigma/v_n]$ as an equivalent of σ. Note that for any variable x, $x\sigma$ is defined, no matter whether x comes from the domain of σ or not, but in the latter case, $x\sigma = x$. Given two substitutions σ and ς whose domains are V and Z respectively, $\sigma\varsigma$ is the substitution with the domain $V \cup Z$, sending every $x \in V \cup Z$ to $(x\sigma)\varsigma$. This notably contrasts with the composition of σ and ς as functions.

1.2 Equational many-sorted logic

Equational reasoning provides a uniform basis for studying various languages and logics about computational effects and beyond. On the simplest level it encompasses the case of so-called *algebraic theories*, i.e. theories where the only thing one can do with functions is to compose them. The language of the many-sorted equational logic consists of two ingredients: type system and signature. Let $\mathcal{W}$ denote a set of *basic sorts*. For instance, $\mathcal{W}$ may contain integer sort, boolean sort, etc. Then the generic *type* is defined by the BNF:

$$\mathrm{Type}_{\mathcal{W}} ::= \mathcal{W} \mid 1 \mid \mathrm{Type}_{\mathcal{W}} \times \mathrm{Type}_{\mathcal{W}}$$

A *Signature* Σ consists of a collection of functional symbols f, each of which is equipped with a *typing*: $A \to B$ where $A, B \in \mathrm{Type}_{\mathcal{W}}$. We denote this situation by $f : A \to B \in \Sigma$ or in most cases, simply $f \in \Sigma$ as the typing is usually implied. In contrast to the single-sorted logic (i.e. when $|\mathcal{W}| = 1$) the general case requires more care — the naive approach may lead to inconsistency as it was shown in [GM81]. The standard idea for overcoming this problem is by introducing so-called *variable contexts*. A variable context Γ is a sequence of the form $(x_1 : A_1, \dots, x_n : A_n)$ where all the variables x_i are distinct, and for every i, A_i is the type of x_i. We denote the result of concatenation of two contexts Γ and Δ as Γ, Δ. In particular, $\Gamma, (x : A)$ will denote the extension of Γ with a new variable x of type A. The latter shall also be commonly shortened to: $\Gamma, x : A$. Finally, for every context $\Gamma = (x_1 : A_1, \dots, x_n : A_n)$ we refer by Γ_t and Γ_v to the type part $\bar{A} = (A_1, \dots, A_n)$ and to the variable part $\bar{x} = (x_1, \dots, x_n)$ correspondingly.

A *context permutation* σ is an operator over variable contexts that performs a shuffling of variables, i.e. it sends a variable context $(x_1 : A_1, \ldots, x_n : A_n)$ to a variable context $(x_{\sigma_1} : A_{\sigma_1}, \ldots, x_{\sigma_n} : A_{\sigma_n})$ where $(\sigma_1, \ldots, \sigma_n)$ is a permutation of the naturals from 1 to n. The action of a context permutation σ upon a context Γ shall be denoted in a postfix manner, i.e. $\Gamma\sigma$. A *term in context* is the construction of the form $\Gamma \rhd t : A$, defined by the rules in Fig. 1.1. The type A here shall be sometimes referred to as a *return type* of t. We denote by $\mathcal{T}_\Sigma$ the set of all well-defined terms in context over signature Σ. Given two terms in context $(\Gamma \rhd s : A)$ and $(\Gamma, x : A \rhd t : B)$ the term in context $(\Gamma \rhd t[s/x] : B)$ is well-defined, as expected (see Lemma 1.2). Let $(x_1 : A_1, \ldots, x_n : A_n \rhd t : A)$ be a term in context. It is easily seen by induction over the term complexity that for every t, $\mathrm{Vars}(t) \subseteq \{x_1, \ldots, x_n\}$.

Remark 1.1. In spite of the importance of variable contexts we shall commonly omit them unless they really affect the reasoning. Also the word 'term' shall normally refer to 'term in context' if no special claims are made.

We adopt the following notational convention about the pairing brackets. For any $n > 1$, let $\langle t_1, t_2, \ldots, t_n\rangle$ be the shorthand for $\langle\langle t_1, \ldots, t_{n-1}\rangle, t_n\rangle$. Hence for every such n, $\langle t_1, \ldots, t_n\rangle$ can be considered as a valid term of the type $(\ldots(A_1 \times A_2) \times \ldots \times A_{n-1}) \times A_n$. Also, $f(\langle t_1, \ldots, t_n\rangle)$ shall be shortened down to $f(t_1, \ldots, t_n)$. Note that such treatment of functions of more than one argument makes impossible to derive the arity of a function symbol from its type, e.g. $f(a, b, c)$ may be a ternary function applied to a, b and c or a binary function applied to a and $\langle b, c\rangle$. This observation shows that one needs to be careful when dealing with arities of functional symbols. Given a term t of type $(\ldots(A_1 \times A_2) \times \ldots \times A_{n-1}) \times A_n$ the notation $\mathrm{pr}^n_i(t)$ with $1 \leqslant i \leqslant n$ will refer to t if $i = n = 1$, to $\mathrm{snd}(t)$ if $i = n > 1$ and to $\mathrm{pr}^{n-1}_i(\mathrm{fst}(t))$ in the remaining cases.

According to the previous discussion, every vector $(t_1, \ldots, t_n)$ can be recast to the term $\langle t_1, \ldots, t_n\rangle$ and vice versa, which shall occasionally be used later on. The later presentation can be understood as a kind of normal form of a vector, specifying how precisely the brackets should be put. This convention allows us to extend the cartesian projection and pairing over vectors: $\mathrm{fst}(x_1, \ldots, x_n) := (x_1, \ldots, x_{n-1})$, $\mathrm{snd}(x_1, \ldots, x_n) := x_n$ and $\langle(x_1, \ldots, x_n), (y_1, \ldots, y_m)\rangle := (x_1, \ldots, x_n, y_1, \ldots, y_m)$. A similar policy applies to the products of types. Namely, the sequence of types $(A_1, \ldots, A_n)$ when needed shall be implicitly converted to $(\ldots(A_1 \times A_2) \times \ldots \times A_{n-1}) \times A_n$ and back. We agree that the cartesian product type constructor binds left to right, i.e.

$$A_1 \times \ldots \times A_n = (\ldots(A_1 \times A_2) \times \ldots \times A_{n-1}) \times A_n.$$

Another notion of substitution besides the standard one introduced previously will be necessary for the purposes of term rewriting. In order to introduce it we must first define the notion of *(reduction) context* (not to be confused with variable contexts, introduced above). Roughly, a reduction context is a term containing a hole, marking an

$$\textbf{(var)}\ \frac{x : A \in \Gamma}{\Gamma \rhd x : A} \qquad \textbf{(app)}\ \frac{f : A \to B \in \Sigma \quad \Gamma \rhd t : A}{\Gamma \rhd f(t) : B} \qquad \textbf{(unit)}\ \frac{}{\Gamma \rhd \star : 1}$$

$$\textbf{(pair)}\ \frac{\Gamma \rhd t : A \quad \Gamma \rhd u : B}{\Gamma \rhd \langle t, u \rangle : A \times B} \qquad \textbf{(fst)}\ \frac{\Gamma \rhd t : A \times B}{\Gamma \rhd \mathsf{fst}(t) : A} \qquad \textbf{(snd)}\ \frac{\Gamma \rhd t : A \times B}{\Gamma \rhd \mathsf{snd}(t) : A}$$

FIGURE 1.1: Term construction rules of the equational logic.

empty position that can be filled by some term or some other context. More formally, a context C is a term over signature $\Sigma \cup \{\Box_{A,B} \mid A, B \in \mathrm{Type}_W\}$ such that at most one of the symbols $\Box_{A,B}$, called *holes* occur in C. Given a term context C with a hole $\Box_{\Gamma_t,B}$ and a term $(\Gamma \rhd t : B)$ we denote by $C\{t\}$ the term, obtained from C by replacing the hole in C with t. In the same way we can place a context C into another context D. The result is evidently again a context, and we denote it by $D\{C\}$. Note that by definition, a context C may not contain a hole at all. If C is a context of this kind, i.e. essentially C is a term we have $C\{t\} \doteq C$ for any t. In contrast to single-sorted settings (cf. e.g. [BN98]) where one symbol for the hole $\Box$ suffices, in multi-sorted cases the typing information should, strictly speaking, be maintained, but since it does not essentially affect the reasoning we shall commonly omit the subscripts at $\Box$.

Variable contexts allow viewing terms as compound functional symbols: the context part prescribes the order of the arguments, and the variables, occurring in the term part, serve as placeholders for the arguments. Intuitively, one would expect that reordering the variables in a variable context should not prevent a term from being well-formed, nor should the extension of a variable context by a dummy variable. This is justified by the following lemma, proved in [Cro94].

Lemma 1.2. *The following term-construction rules are admissible with respect to the system in Fig. 1.1.*

$$\frac{\Gamma \rhd t : A}{\Gamma\sigma \rhd t : A} \qquad \frac{\Gamma \rhd t : A}{\Gamma, \Delta \rhd t : A} \qquad \frac{\Gamma \rhd t : A \quad \Gamma, x : A \rhd s : B}{\Gamma \rhd s[t/x] : B}$$

where σ is an arbitrary context permutation.

A proof system for the many-sorted equation logic EQ operates with units, called *equations in context* having form $\Gamma \rhd t = s : A$, where $\Gamma \rhd t : A$ and $\Gamma \rhd s : A$ are some terms in context. Given a set of axioms Φ, each of which is an equation in context, $\Gamma \rhd t = s : A$ is provable iff it can be derived from Φ by the rules in Fig. 1.2 (cf. [AKKB99]). This shall be denoted by $\Phi \vdash_{\mathsf{EQ}} t = s$, or simply $\vdash_{\mathsf{EQ}} p = q$ if $\Phi = \emptyset$.

The rules **(refl)**, **(sym)** and **(trans)** are the standard rules, reflecting the properties of the equivalence operator: reflexivity, transitivity and symmetry. The generic congruence rule **(cong)** together with the instantiation rule **(inst)** completely characterise the

$$\textbf{(refl)}\ \frac{\Gamma \triangleright s : A \in \mathfrak{T}_\Sigma}{\Gamma \triangleright s = s : A} \qquad \textbf{(sym)}\ \frac{\Gamma \triangleright s = t : A}{\Gamma \triangleright t = s : A} \qquad \textbf{(trans)}\ \frac{\Gamma \triangleright s = t : A \quad \Gamma \triangleright t = r : A}{\Gamma \triangleright s = r : A}$$

$$\textbf{(cong)}\ \frac{f : A \to B \in \Sigma \quad \Gamma \triangleright s = t : A}{\Gamma \triangleright f(s) = f(r) : B} \qquad \textbf{(pair\_cong)}\ \frac{\Gamma \triangleright s = p : A \quad \Gamma \triangleright t = q : B}{\Gamma \triangleright \langle s, t\rangle = \langle p, q\rangle : A \times B}$$

$$\textbf{(unit)}\ \frac{\Gamma \triangleright s : 1 \in \mathfrak{T}_\Sigma \quad \Gamma \triangleright t : 1 \in \mathfrak{T}_\Sigma}{\Gamma \triangleright s = t : 1} \qquad \textbf{(pair)}\ \frac{\Gamma \triangleright t : A \times B \in \mathfrak{T}_\Sigma}{\Gamma \triangleright \langle \mathsf{fst}(t), \mathsf{snd}(t)\rangle = t : A \times B}$$

$$\textbf{(fst)}\ \frac{\Gamma \triangleright s : A \in \mathfrak{T}_\Sigma \quad \Gamma \triangleright t : B \in \mathfrak{T}_\Sigma}{\Gamma \triangleright \mathsf{fst}\langle s, t\rangle = s : A} \qquad \textbf{(fst\_cong)}\ \frac{\Gamma \triangleright s = t : A \times B}{\Gamma \triangleright \mathsf{fst}(s) = \mathsf{fst}(t) : A}$$

$$\textbf{(snd)}\ \frac{\Gamma \triangleright s : A \in \mathfrak{T}_\Sigma \quad \Gamma \triangleright t : B \in \mathfrak{T}_\Sigma}{\Gamma \triangleright \mathsf{snd}\langle s, t\rangle = t : B} \qquad \textbf{(snd\_cong)}\ \frac{\Gamma \triangleright s = t : A \times B}{\Gamma \triangleright \mathsf{snd}(s) = \mathsf{snd}(t) : B}$$

$$\textbf{(inst)}\ \frac{\Gamma \triangleright p : A \in \mathfrak{T}_\Sigma \quad \Gamma, x : A \triangleright t = s : B}{\Gamma \triangleright t[p/x] = s[p/x] : B}$$

FIGURE 1.2: EQ: Proof calculus of the many-sorted equational logic.

equality predicate. The remaining rules are to specify the behaviour of cartesian primitives.

Context manipulations now appear as admissible rules.

Theorem 1.3 (cf. e.g. [Cro94]). *The derivation rules*

$$\textbf{(weak)}\ \frac{\Gamma \triangleright t = s : A}{\Delta, \Gamma \triangleright t = s : A} \qquad \textbf{(perm)}\ \frac{\Gamma \triangleright t = s : A}{\Gamma\sigma \triangleright t = s : A}$$

are admissible with respect to EQ. *Here σ is an arbitrary context permutation.*

Proof. The proof of **(weak)** immediately follows from the observation that in every rule of EQ the variable contexts both in the premises and in the conclusion can be soundly extended to the left by Δ. For those rules which contain as premises statements about the existence of terms in context, we call Lemma 1.2 in order to ensure that once $(\Gamma \triangleright p : A)$ is a well-formed term, then $(\Delta, \Gamma \triangleright p : A)$ is also a well-formed term.

The proof of **(perm)** is essentially similar, but observe that due to **(inst)** the derivation for $(\Gamma \triangleright t = s : A)$ might contain variable contexts, extending Γ to the right. We extend σ over such term contexts by putting $(\Gamma, \Delta)\sigma := (\Gamma\sigma, \Delta)$. It is clear that for every rule from Fig. 1.2, if we apply σ both to the premises and to the conclusion we again obtain an instance of a rule from Fig. 1.2. Again, for those premises which only state whether terms are well-formed, this is justified by Lemma 1.2. □

Lemma 1.4. *Given some fixed set of axioms Φ, the instantiation rule* **(inst)** *can be replaced by*

$$\textbf{(inst}_\Phi\textbf{)}\ \frac{\Gamma \rhd t_i : A_i \in \mathcal{T}_\Sigma \quad (x_1 : A_1, \ldots x_n : A_n) \rhd t = s : A \in \Phi}{\Gamma \rhd t[t_1/x_1, \ldots, t_n/x_n] = s[t_1/x_1, \ldots, t_n/x_n] : A}$$

equivalently in the following sense: if EQ_Φ *is the proof system, obtained from* EQ *by such a replacement then* $\Phi \vdash_{\mathsf{EQ}} p = q$ *iff* $\vdash_{\mathsf{EQ}_\Phi} p = q$*. In particular, if the set of axioms* Φ *is empty, then rule* **(inst)** *can be eliminated from any proof* $\Phi \vdash_{\mathsf{EQ}} p = q$*.*

Proof. Suppose we have a proof of $\vdash_{\mathsf{EQ}_\Phi} p = q$. Then it can be converted into a proof of $\Phi \vdash_{\mathsf{EQ}} p = q$ by replacing every fragment of the form

$$\frac{\Gamma \rhd t_i : A_i \in \mathcal{T}_\Sigma \quad (x_1 : A_1, \ldots x_n : A_n) \rhd t = s : A \in \Phi}{\Gamma \rhd t[t_1/x_1, \ldots, t_n/x_n] = s[t_1/x_1, \ldots, t_n/x_n] : A}$$

by the following series of **(inst)** as follows:

$$\frac{\Gamma \rhd t_n : A_n \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma \rhd t_{n-1} : A_{n-1} \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma \rhd t_1 : A_1 \in \mathcal{T}_\Sigma \quad (x_1 : A_1, \ldots x_n : A_n) \rhd t = s : A}{\vdots}}{\Gamma, x_n : A_n \rhd t[t_1/x_1, \ldots, t_{n-1}/x_{n-1}] = s[t_1/x_1, \ldots, t_{n-1}/x_{n-1}] : A}}{\Gamma \rhd t[t_1/x_1, \ldots, t_n/x_n] = s[t_1/x_1, \ldots, t_n/x_n] : A}$$

Let us prove the equivalence in the converse direction: provided we have a proof of $\Phi \vdash_{\mathsf{EQ}} p = q$, we show how it can be converted into a proof of $\vdash_{\mathsf{EQ}_\Phi} p = q$. Essentially, we need to show that **(inst)** is admissible in EQ_Φ. For convenience, we generalise this rule a little, so that it captures simultaneous instantiations of many variables:

$$f\overline{\textbf{(inst)}}\ \frac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \quad \Gamma, x_1 : A_1 \ldots, x_n : A_n \rhd t = s : B}{\Gamma \rhd t[r_1/x_1, \ldots, r_n/x_n] = s[r_1/x_1, \ldots, r_n/x_n] : B}$$

We proceed by induction over the complexity of the proof of the premise $\Gamma, \bar{x} : \bar{A} \rhd t = s : B$ of $\overline{\textbf{(inst)}}$. As the induction base we have $(\Gamma, \bar{x} : \bar{A} \rhd t = s : B \in \Phi)$ in which case the rule is acceptable, since it becomes an instance of **(inst$_\Phi$)**. Let us prove the induction step. The equation $\Gamma, \bar{x} : \bar{A} \rhd t = s : B$ should match the conclusion of some other rule of EQ. Consider only the most typical cases and drop the remaining rules, as they can be proved analogously.

The premise is obtained by **(fst)**. This restricts the form of t and s down to $\mathrm{fst}\langle r, q\rangle$ and r correspondingly and we have the derivation:

$$\frac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma, \bar{x} : \bar{A} \rhd r : B \in \mathcal{T}_\Sigma \quad \Gamma, \bar{x} : \bar{A} \rhd q : C \in \mathcal{T}_\Sigma}{\Gamma, \bar{x} : \bar{A} \rhd \mathrm{fst}\langle r, q\rangle = r : B}\,\textbf{(fst)}}{\Gamma \rhd \mathrm{fst}\langle r[\bar{r}/\bar{x}], q[\bar{r}/\bar{x}]\rangle = r[\bar{r}/\bar{x}] : B}\,\overline{\textbf{(inst)}}$$

which is equivalent to

$$\frac{\Gamma \rhd r[\bar{r}/\bar{x}] : B \in \mathcal{T}_\Sigma \quad \Gamma \rhd q[\bar{r}/\bar{x}] : C \in \mathcal{T}_\Sigma}{\Gamma \rhd \mathsf{fst}\langle r[\bar{r}/\bar{x}], q[\bar{r}/\bar{x}]\rangle = r[\bar{r}/\bar{x}] : B}\ \textbf{(fst)}$$

and thus $\textbf{(}\overline{\textbf{inst}}\textbf{)}$ is completely eliminated.

The premise is obtained by **(trans)**. Let the premises of **(trans)** be $\Gamma, \bar{x} : \bar{A} \rhd t = r : B$ and $\Gamma, \bar{x} : \bar{A} \rhd r = s : B$, i.e. we have:

$$\frac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma, \bar{x} : \bar{A} \rhd t = r : B \quad \Gamma, \bar{x} : \bar{A} \rhd r = s : B}{\Gamma, \bar{x} : \bar{A} \rhd t = s : B}\ \textbf{(trans)}}{\Gamma \rhd t[\bar{r}/\bar{x}] = s[\bar{r}/\bar{x}] : B}\ \textbf{(}\overline{\textbf{inst}}\textbf{)}$$

This can be equivalently replaced by

$$\frac{\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \ \ \Gamma, \bar{x} : \bar{A} \rhd t = r : B}{\Gamma \rhd t[\bar{r}/\bar{x}] = r[\bar{r}/\bar{x}] : B}\ \textbf{(}\overline{\textbf{inst}}\textbf{)} \quad \dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \ \ \Gamma, \bar{x} : \bar{A} \rhd r = s : B}{\Gamma \rhd r[\bar{r}/\bar{x}] = s[\bar{r}/\bar{x}] : B}\ \textbf{(}\overline{\textbf{inst}}\textbf{)}}{\Gamma \rhd t[\bar{r}/\bar{x}] = s[\bar{r}/\bar{x}] : B}\ \textbf{(trans)}$$

and the newly introduced applications of $\textbf{(}\overline{\textbf{inst}}\textbf{)}$ can be eliminated by induction hypothesis.

The premise is obtained by $\textbf{(}\overline{\textbf{inst}}\textbf{)}$*.* Without going too much into details, observe that two applications of $\textbf{(}\overline{\textbf{inst}}\textbf{)}$ in raw can always be merged down into one. Then we are done by induction hypothesis. □

The most important admissible rule of EQ is the *substitution rule*:

$$\textbf{(subst)}\ \frac{\Gamma \rhd s = t : A \quad \Gamma, x : A \rhd p = q : B}{\Gamma \rhd p[s/x] = q[t/x] : B}$$

Theorem 1.5. *The rule* **(subst)** *is admissible in* EQ.

Proof. Let us first prove by induction over the term complexity of p that the rule

$$\textbf{(cong')}\ \frac{\Gamma, x : A \rhd B \in \mathcal{T}_\Sigma \quad \Gamma \rhd s = t : A}{\Gamma \rhd p[s/x] = p[t/x] : B}$$

is admissible in EQ. In the base case either $p = x$, and hence **(cong')** holds trivially, or p does not contain x (if p is a constant or a variable different from x) and therefore the conclusion of **(cong')** turns into a trivial identity, provable by **(refl)**. Suppose, p is obtained by one of the term-construction rules. For instance, if p is obtained by **(app)**, there should exist $f \in \Sigma$ and q such that $p = f(q)$. By induction hypothesis,

$$\{\Gamma \rhd s = t : A\} \vdash_{\mathsf{EQ}} \Gamma \rhd q[s/x] = q[t/x] : B,$$

$$\textbf{(var)}\ \frac{\Gamma = (x_1 : A_1, \ldots, x_n : A_n)}{[\![\Gamma \rhd x_i : A_i]\!] = \pi_i^n} \qquad \textbf{(app)}\ \frac{f : A \to B \in \Sigma \quad [\![\Gamma \rhd t : A]\!] = h}{[\![\Gamma \rhd f(t) : B]\!] = [\![f]\!] \circ h}$$

$$\textbf{(unit)}\ \frac{}{[\![\Gamma \rhd \star : 1]\!] = !_{[\![\Gamma]\!]}} \qquad \textbf{(pair)}\ \frac{[\![\Gamma \rhd t : A]\!] = h \quad [\![\Gamma \rhd s : B]\!] = g}{[\![\Gamma \rhd \langle t, s\rangle : A \times B]\!] = \langle h, g\rangle}$$

$$\textbf{(fst)}\ \frac{[\![\Gamma \rhd t : A \times B]\!] = h}{[\![\Gamma \rhd \mathsf{fst}(t) : A]\!] = \pi_1 \circ h} \qquad \textbf{(snd)}\ \frac{[\![\Gamma \rhd t : A \times B]\!] = h}{[\![\Gamma \rhd \mathsf{snd}(t) : A]\!] = \pi_2 \circ h}$$

FIGURE 1.3: Interpretation of terms of the equational logic.

from which we obtain **(cong′)** by **(cong)**. The case of either of the rules **(unit)**, **(pair)**, **(fst)** and **(snd)** is studied analogously.

Now observe that the substitution rule can be presented as the composition

$$\dfrac{\dfrac{\Gamma, x : A \rhd p : B \in \mathcal{T}_\Sigma \quad \Gamma \rhd s = t : A}{\Gamma \rhd p[s/x] = p[t/x] : B}\,\textbf{(cong′)} \qquad \dfrac{\Gamma \rhd t : A \in \mathcal{T}_\Sigma \quad \Gamma, x : A \rhd p = q : B}{\Gamma \rhd p[t/x] = q[t/x] : B}\,\textbf{(inst)}}{\Gamma \rhd p[s/x] = q[t/x] : B}\,\textbf{(trans)}$$

which completes the proof of the theorem. □

Suppose every sort $A \in \mathcal{W}$ is put into correspondence with some object $[\![A]\!]$ of a cartesian category **C**, i.e. a category possessing finite products (hence also the terminal object as a product of zero elements). Then we can extend the interpretation of types over all $\mathrm{Type}_{\mathcal{W}}$: $[\![1]\!] := 1$, $[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$. Furthermore, this can be extended to term contexts by defining: $[\![x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n]\!] := [\![A_1]\!] \times [\![A_2]\!] \times \ldots \times [\![A_n]\!]$. Provided that every $f : A \to B \in \Sigma$ is interpreted as a morphism $[\![f]\!] \in \mathsf{Hom}_{\mathbf{C}}([\![A]\!], [\![B]\!])$, this interpretation can be extended so as to assign to every term in context $(\Gamma \rhd t : A)$ a morphism $[\![t]\!] \in \mathsf{Hom}_{\mathbf{C}}([\![\Gamma]\!], [\![A]\!])$ according to Fig. 1.3. Here, for every $A, B, C \in \mathrm{Type}_{\mathcal{W}}^{T}$, $!_A \in \mathsf{Hom}(A, 1)$ is the terminal morphism, $\pi_1 \in \mathsf{Hom}(A \times B, A)$, $\pi_2 \in \mathsf{Hom}(A \times B, B)$ are cartesian projections, $\langle f, g\rangle \in \mathsf{Hom}(A, B \times C)$ is the cartesian pairing, defined for every $f \in \mathsf{Hom}(A, B)$ and $g \in \mathsf{Hom}(A, C)$. The family of morphisms π_i^n, with $1 \leqslant i \leqslant n$ is recursively defined as follows: $\pi_1^1 := \mathrm{id}$, $\pi_{n+1}^{n+1} := \pi_2$ and $\pi_i^{n+1} := \pi_i^n \circ \pi_1$ $(i \leqslant n)$.

Equational many-sorted logic is the internal language of cartesian categories. This is justified by the following theorem (cf. e.g. [Cro94]).

Theorem 1.6. *Equational many-sorted logic is sound and strongly complete over cartesian categories under the interpretation, given in Fig. 1.2.*

Proof. As usual, soundness is easily verified by considering all the rules one by one and making sure that all of them indeed hold over cartesian categories. In order to prove strong completeness one needs first to fix a set of axioms Φ and then proceed with the construction of a term category $\mathbf{C}_{\Sigma,\Phi}$ validating those and only those equalities which can be derived from Φ. The objects of $\mathbf{C}_{\Sigma,\Phi}$ are types from $\mathrm{Type}_{\mathcal{W}}$ and morphisms $\mathsf{Hom}_{\mathbf{C}}(A,B)$ are terms in context $(x : A \rhd t : B)$ modulo the provable equivalence in EQ which we denote by $\sim$. We interpret every base type $A \in \mathcal{W}$ trivially: $[\![A]\!] := A$ and every function symbol $f : A \to B \in \Sigma$ by assigning: $[\![f]\!] := [x : A \rhd f(x) : B]_{\sim}$. It easily follows from the definition that the semantic brackets extended over terms satisfy the equation:

$$[\![x_1 : A_1, \ldots, x_n : A_n \rhd t : B]\!] = [x : (A_1 \times \ldots \times A_n) \rhd t[\mathrm{pr}^n_1(x)/x_1, \ldots, \mathrm{pr}^n_n(x)/x_n] : B]_{\sim}.$$

This immediately ensures the crucial property of $\mathbf{C}_{\Sigma,\Phi}$: any two terms $(\Gamma \rhd p : A)$ and $(\Gamma \rhd q : A)$ have the same interpretation over it iff $\Phi \vdash_{\mathsf{EQ}} \Gamma \rhd p = q : A$.

We note that the main point of the construction of $\mathbf{C}_{\Sigma,\Phi}$ is the definition of the composition operator as term substitution:

$$[y : B \rhd q : C]_{\sim} \circ [x : A \rhd p : B]_{\sim} := [x : A \rhd q[p/y] : C]_{\sim}. \tag{1.1}$$

The identity morphisms $\mathrm{id}_A \in \mathsf{Hom}(A,A)$ are defined to be $[A : x \rhd x : A]_{\sim}$. The detailed verification of the axioms of the cartesian category is found in [Cro94]. □

1.3 Monads and the metalanguage of effects

We are now positioned to introduce the notion of the monad. This notion goes back to the core of category theory. Its generic meaning is very abstract – basically it captures some closure and compositionality properties of a wide range of various algebraic constructions. It turned out that constructions of this kind arise rather commonly in mathematics, and as a result, the concept of the monad proved to be highly appropriate. Our interest in the notion of the monad is of course from the perspective of their computational interpretation. Monads whose intended meaning is to model computational effects are commonly referred to as *computational monads*. The motivation behind this term is of course purely methodological, justified by the fact that not all the results about monads admit a reasonable computational interpretation. In this section we recall the ones which do and which are relevant for further presentation.

Definition 1.7 (Monad, cf. e.g. [ML71])**.** A monad $\mathbb{T}$ over a category $\mathbf{C}$ is an endofunctor $T : \mathbf{C} \to \mathbf{C}$ augmented with two natural transformations $\eta : I \to T$ and $\mu : T^2 \to T$, called *unit* and *multiplication*, which make the following diagrams commute

$$\begin{tikzcd}
T^3 \arrow[r, "T\mu"] \arrow[d, "\mu T"'] & T^2 \arrow[d, "\mu"] \\
T^2 \arrow[r, "\mu"] & T
\end{tikzcd}
\qquad
\begin{tikzcd}
IT \arrow[r, "\eta T"] \arrow[d, equal] & T^2 \arrow[d, "\mu"] & TI \arrow[l, "T\eta"'] \arrow[d, equal] \\
T \arrow[r, equal] & T \arrow[r, equal] & T
\end{tikzcd}$$

Here I denotes the identity functor.

Definition 1.8 (Kleisli category, cf. e.g. [Man76]). A *Kleisli triple* $\mathbb{T}$ over category $\mathbf{C}$ is given by the following data.

- An endomap T over $\mathrm{Ob}(\mathbf{C})$.
- A morphism $\eta_A : A \to TA$ for every $A \in \mathrm{Ob}(\mathbf{C})$, called *unit*.
- Unary operation $\_^\dagger$, called *Kleisli star*, taking every morphism $f : A \to TB$ to $f^\dagger : TA \to TB$.

Moreover, the equations

$$\eta_A^\dagger = \mathrm{id}_{TA}, \qquad f^\dagger \circ \eta_A = f, \qquad g^\dagger \circ f^\dagger = (g^\dagger \circ f)^\dagger \tag{1.2}$$

are satisfied. We refer to the defined Kleisli triple as $(T, \eta, \_^\dagger)$.

Definitions 1.7 and 1.8 partly share terms and notation, but this clash is resolved by the fact of their equivalence: every monad $\mathbb{T}$ in the sense of Definition 1.7 gives rise to a Kleisli triple by defining a Kleisli star according to the assignment:

$$f^\dagger := \mu \circ Tf. \tag{1.3}$$

Conversely, given a Kleisli triple $\mathbb{T} = (T, \eta, \_^\dagger)$ define

$$Tf := (\eta \circ f)^\dagger, \qquad \mu := \mathrm{id}^\dagger. \tag{1.4}$$

The detailed proof of the equivalence of Definition 1.7 and 1.8 can be found in [Man76]. Definition 1.8 is rather more appealing from the computational point of view (and it shall be given preference henceforth) because it provides better intuition, based on programming experience. Roughly, TA denotes the type of computations with results in A. The elements of TA can be understood as *computations*, i.e. programs which may (or may not, e.g. if the computation does not terminate) be evaluated to elements of type A. Then η_A is an injection (not necessarily in categorical sense) of a value a into a trivial computation, i.e. the one which always evaluates to a. Finally, the Kleisli star coherently lifts any function whose return type is computational to a function whose argument is also of a computational type. Essentially, this means the mapping information about every single value of A is sufficient to tell how the computations of type

TA should be handled. This gives an idea how any two functions $f : A \to TB$ and $g : B \to TC$ can be composed together. Specifically, let us put by definition

$$g \diamond f := g^{\dagger} \circ f. \tag{1.5}$$

This is known as *Kleisli composition*, and in fact, morphisms whose targets are images of T form a category $\mathbf{C}_{\mathbb{T}}$ with Kleisli composition as the composition of morphisms and η as the identity morphism. So the defined category $\mathbf{C}_{\mathbb{T}}$ is known as the *Kleisli category* of $\mathbb{T}$.

Abstract speculations about the computational view of monads are justified by the following standard examples of computationally relevant monads.

Example 1.1. [Mog91] Provided that the underlying category $\mathbf{C}$ has enough structure, any the following is an example of a monad $\mathbb{T} = (T, \eta, \_^{\dagger})$ over $\mathbf{C}$.

1. *Identity (or trivial) monad:* $TA = A$, $\eta_A = \mathrm{id}_A$, $f^{\dagger} = f$.
2. *Exception monad:* $TA = A + E$,
   $\eta_A = \mathrm{inl}$, $f^{\dagger} = [f, \mathrm{inr}]$ with E presenting a pool of exceptions.
3. *Powerset monad:* $TA = \mathcal{P}A$,
   $\eta_A = \lambda a.\{a\}$, $f^{\dagger} = \lambda c.\{f(x) \mid x \in c\}$ where $\mathcal{P}$ is a covariant powerset functor.
4. *State monad:* $TA = S \to (A \times S)$, $\eta_A = \lambda a.\lambda s.\langle a, s\rangle$,
   $f^{\dagger} = \lambda c.\, \lambda s.\, \big(\lambda\langle x, y\rangle.\, f(x)(y)\big)$ with S presenting the states.
5. *Interactive input:* $TA = \mu X.(A + (I \to X))$, $\eta_A = \mathrm{inl}$,
   $f^{\dagger} = \mu g.\, [f, \lambda h.\, \lambda u.\, g(h(u))]$ with I being an object presenting the input stream.
6. *Interactive output:* $TA = \mu X.(A + (O \times X))$, $\eta_A = \mathrm{inl}$,
   $f^{\dagger} = \mu g.\, \big[f, \lambda\langle u, x\rangle.\, \langle u, g(x)\rangle\big]$ with O being an object presenting the output stream.
7. *Continuation monad:* $TA = (A \to R) \to R$, $\eta_A = \lambda a.\, \lambda r.\, r(a)$,
   $f^{\dagger} = \lambda c.\lambda r.c(\lambda a.f(a)(r))$, with R being the type of global outcomes.

The monads (1) – (7) are basic ones. They can be used as building blocks in order to obtain more complex monads, modelling more advanced computational effects. Here are several interesting ones.

8. *Nondeterministic state monad:* $TA = S \to \mathcal{P}(A \times S)$,
9. *Input-output monad:* $TA = \mu X.(A + (I \to O \times X))$,
10. *Java monad* [JP03]: $TA = S \to (A \times S + E \times S + 1)$.

Now the notion of the Kleisli category can be exemplified as follows. Let $\mathbb{T}$ be the exception monad over **Set** with $E = 1$. Then the Kleisli category $\mathbf{C}_\mathbb{T}$ precisely captures the category of strict partial maps relative to the underlying category whose morphisms are total functions [Mog91]. Another well-recognised example of this sort is obtained from the powerset monad: provided the underlying category is the category of sets and functions, the Kleisli category is the category of sets and relations. In general, we interpret the Kleisli category of a monad as a category of effectful programs with respect to the underlying category whose morphisms should be considered as effectless programs.

One of the first categorical results about monads was that any adjunction gives rise to a monad. An important property of the Kleisli category $\mathbf{C}_\mathbb{T}$ is that it is related to $\mathbf{C}$ by an adjunction, and conversely this adjunction gives rise to the monad that is precisely $\mathbb{T}$.

Definition 1.9 (cf. e.g. [Mog89a]). *Kleisli construction* takes a monad $\mathbb{T} = (T, \eta, \_^\dagger)$ over some category $\mathbf{C}$ as input and produces its Kleisli category $\mathbf{C}_\mathbb{T}$ together with the adjunction $(K_\mathbb{T}, G_\mathbb{T}, \eta, \epsilon)$ whose components are defined as follows.

- $K_\mathbb{T} : \mathbf{C} \to \mathbf{C}_\mathbb{T}$, the left adjoint functor sends every $A \in \mathsf{Ob}\,(\mathbf{C})$ to $A \in \mathsf{Ob}\,(\mathbf{C}_\mathbb{T})$ and every $f \in \mathsf{Hom}_\mathbf{C}(A, B)$ to $\eta_B \circ f \in \mathsf{Hom}_{\mathbf{C}_\mathbb{T}}(A, B)$.
- $G_\mathbb{T} : \mathbf{C}_\mathbb{T} \to \mathbf{C}$, the right adjoint functor sends every $A \in \mathsf{Ob}\,(\mathbf{C}_\mathbb{T})$ to $TA \in \mathsf{Ob}\,(\mathbf{C})$ and every $f \in \mathsf{Hom}_{\mathbf{C}_\mathbb{T}}(A, B) = \mathsf{Hom}_\mathbf{C}(A, TB)$ to $f^\dagger \in \mathsf{Hom}_\mathbf{C}(TA, TB)$.
- $I_\mathbf{C} \to G_\mathbb{T} K_\mathbb{T}$, the unit of the adjudication is the unit of monad $\mathbb{T} = G_\mathbb{T} K_\mathbb{T}$.
- $\epsilon : G_\mathbb{T} K_\mathbb{T} \to I_{\mathbf{C}_\mathbb{T}}$, the counit of the adjunction is the natural transformation $\epsilon_A = \mathrm{id}_{TA} \in \mathsf{Hom}_\mathbb{T}(TA, A)$.

Theorem 1.10 (cf. e.g. [BW85]). *Let $(K_\mathbb{T}, G_\mathbb{T}, \eta, \epsilon)$ be the adjunction between* $\mathbf{C}$ *and* $\mathbf{C}_\mathbb{T}$, *provided by the Kleisli construction. Then $(G_\mathbb{T} K_\mathbb{T}, \eta, G_\mathbb{T} \epsilon K_\mathbb{T})$ is a monad, and, moreover, it coincides with the original monad $\mathbb{T} = (T, \eta, \mu)$.*

Another classical construction generating monads from adjunctions is well-known among category theorists: the *Eilenberg-Moore construction* (cf. e.g. [ML71]). From the computational point of view, it is far less significant than Kleisli construction since the latter has a very sensible and clear computational interpretation: the functor $K_\mathbb{T}$ injects effectless functions into the realm of effectful ones.

A typical underlying category for a monad is **Set**. But most of the monads provided by Example 1.1 make sense under more general settings. For example, one can define an exception monad over any category, possessing finite co-products, or a state monad over any cartesian closed category, etc. Nevertheless, there is a construction transforming a monad over any category to a monad over a category almost as reach as **Set** (Theorem 1.37).

Definition 1.11. Let **C** be a cartesian category. A monad $\mathbb{T} = (T, \eta, \mu)$ over **C** is *strong* if it is equipped with a natural transformation

$$\tau_{A,B} : A \times TB \to T(A \times B)$$

called *(tensorial) strength* making the following diagrams commute.

$$\begin{tikzcd}
1 \times TA \arrow[r, "\tau_{1,A}"] \arrow[d, "\pi_2"'] & T(1 \times A) \arrow[d, "T\pi_2"] \\
TA \arrow[r, equal] & TA
\end{tikzcd}
\qquad
\begin{tikzcd}
(A \times B) \times TC \arrow[rr, "\tau_{A\times B,C}"] \arrow[d, "\alpha_{A,B,TC}"'] & & T((A \times B) \times C) \arrow[d, "T\alpha_{A,B,C}"] \\
A \times (B \times TC) \arrow[r, "\mathrm{id}_A \times \tau_{B,C}"] & A \times T(B \times C) \arrow[r, "\tau_{A,B\times C}"] & T(A \times (B \times C))
\end{tikzcd}$$

$$\begin{tikzcd}
A \times B \arrow[r, "\mathrm{id}_{A\times B}"] \arrow[d, "\mathrm{id}_A \times \eta_B"'] & A \times B \arrow[d, "\eta_{A\times B}"] \\
A \times TB \arrow[r, "\tau_{A,B}"] & T(A \times B)
\end{tikzcd}
\qquad
\begin{tikzcd}
A \times T^2B \arrow[r, "\tau_{A\times TB}"] \arrow[d, "\mathrm{id}_A \times \mu_B"'] & T(A \times TB) \arrow[r, "T\tau_{A,B}"] & T^2(A \times B) \arrow[d, "\mu_{A\times B}"] \\
A \times TB \arrow[rr, "\tau_{A,B}"] & & T(A \times B)
\end{tikzcd}$$

Here α refers to product associativity natural isomorphism. We shall use the notation (T, η, μ, τ) for the strong monad $\mathbb{T} = (T, \eta, \mu)$ with the strength τ and $(T, \eta, \_^\dagger, \tau)$ for the corresponding Kleisli triple.

Remark 1.12. In case if the underlying category **C** is cartesian closed, the strength is known to be equivalent to enrichment of the monad functor over **C** [Koc72] (**C** is enriched over itself because of cartesian closeness). In particular, this means that every monad over **Set** is strong (because every endofunctor over **Set** is enriched over **Set**).

The language of commutative diagrams might become cumbersome when it comes to really involved calculations. This is why for reasoning about strong monads it shall be overwhelmingly simulated by its flat counterpart (i.e. by unfolding every diagram into a series of equations between morphism chains, connected by the composition operator). A rather more advantageous language, called the metalanguage of effects, will be introduced later. Still, many calculations call for identities of the more primitive level (including those which are needed to prove correctness of the metalanguage of effects). We summarise for reference some useful properties of this kind in the following lemma.

For the remainder of this thesis we agree that the composition operator $\circ$ binds more strongly than Kleisli composition $\diamond$. For the sake of brevity and readability we commonly omit $\circ$ in the evident places, such as $\eta \circ \pi_1$ and $\tau_{A,B} \circ \langle \mathrm{id}, \eta \rangle$.

Lemma 1.13. *In every strong monad $\mathbb{T} = (T, \eta, \_^\dagger, \tau)$ the following identities hold:*

$$T\pi_2 \circ \tau_{A,B} = \pi_2, \tag{1.6}$$

$$\tau_{A,B}(\mathrm{id} \times \eta) = \eta, \tag{1.7}$$

$$\tau_{A,C}\langle h, f \diamond \tau_{A,B}\langle h, g\rangle\rangle = \tau_{A,C}\langle \pi_1, f\rangle \diamond \tau_{A,B}\langle h, g\rangle, \tag{1.8}$$

$$\tau_{A,C}(\mathrm{id} \times f \diamond g) = \tau_{A,C}(\mathrm{id} \times f) \diamond \tau_{A,B}(\mathrm{id} \times g). \tag{1.9}$$

Proof. Let us prove the identities in question one by one.

Equation (1.6). By the naturality of the strength $T\pi_2 \circ \tau_{A,B} = T\pi_2 \circ T(! \times \mathrm{id}) \circ \tau_{A,B}$ is equal to $T\pi_2 \circ \tau_{1,B} \circ (! \times \mathrm{id})$. By the left upper diagram from the definition of strength this is equal to $\pi_2 \circ (! \times \mathrm{id})$ i.e. to π_2.

Equation (1.7) is precisely the string form of the left bottom diagram from the definition of strength.

Equation (1.8). First let us prove the partial case: $f := \eta$ and $C := A \times B$. The identity in question thus simplifies down to:

$$\tau_{A,C}\langle h, \tau_{A,B}\langle h, g\rangle\rangle = \tau_{A,C}\langle \pi_1, \eta\rangle \diamond \tau_{A,B}\langle h, g\rangle. \tag{1.10}$$

Observe that the right top diagram from the definition of the strength amounts to the identity:

$$\tau_{A,C}\langle \pi_1\pi_1, \tau_{A,B}\langle \pi_2\pi_1, \pi_2\rangle\rangle = T(\pi_1\pi_1, \langle \pi_2\pi_1, \pi_2\rangle) \circ \tau_{A\times A,B}.$$

By composing both sides of it with $\langle\langle h, h\rangle, g\rangle$ on the right we obtain:

$$\begin{aligned}
&\tau_{A,C}\langle h,\tau_{A,B}\langle h, g\rangle\rangle \\
&\quad= T\langle \pi_1\pi_1, \langle \pi_2\pi_1, \pi_2\rangle\rangle \circ \tau_{A\times A,B}\langle\langle h, h\rangle, g\rangle \\
&\quad= T\langle \pi_1\pi_1, \langle \pi_2\pi_1, \pi_2\rangle\rangle \circ \tau_{A\times A,B}(\langle \mathrm{id}, T\,\mathrm{id}\rangle \times \mathrm{id}) \circ \langle h, g\rangle \\
&\quad= T\langle \pi_1\pi_1, \langle \pi_2\pi_1, \pi_2\rangle\rangle \circ T(\langle \mathrm{id}, \mathrm{id}\rangle \times \mathrm{id}) \circ \tau_{A,B}\langle h, g\rangle && \text{[by nat. of } \tau\text{]} \\
&\quad= T\langle \pi_1, \mathrm{id}\rangle \circ \tau_{A,B}\langle h, g\rangle \\
&\quad= \mu \circ T\eta \circ T\langle \pi_1, \mathrm{id}\rangle \circ \tau_{A,B}\langle h, g\rangle && \text{[by Def. 1.7]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times \eta) \circ T\langle \pi_1, \mathrm{id}\rangle \circ \tau_{A,B}\langle h, g\rangle && \text{[by 1.7]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T\langle \pi_1, \eta\rangle \circ \tau_{A,B}\langle h, g\rangle \\
&\quad= \tau_{A,C}\langle \pi_1, \eta\rangle \diamond \tau_{A,B}\langle h, g\rangle && \text{[by 1.3, 1.5]}
\end{aligned}$$

which proves (1.10). Now the general case is proved as follows:

$$\begin{aligned}
&\tau_{A,C}\langle h,f \diamond \tau_{A,B}\langle h, g\rangle\rangle \\
&\quad= \tau_{A,C}(\mathrm{id} \times \mu) \circ \langle h, Tf \circ \tau_{A,B}\langle h, g\rangle\rangle && \text{[by 1.5]} \\
&\quad= \mu \circ T\tau_{A,C} \circ \tau_{A,TC}(\mathrm{id} \times Tf) \circ \langle h, \tau_{A,B}\langle h, g\rangle\rangle && \text{[by Def. 1.11]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ \tau_{A,A\times B}\langle h, \tau_{A,B}\langle h, g\rangle\rangle && \text{[by nat. of } \tau\text{]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ (\tau_{A,A\times B}\langle \pi_1, \eta\rangle \diamond \tau_{A,B}\langle h, g\rangle) && \text{[by 1.10]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ \\
&\qquad (\tau_{A,A\times B}(\mathrm{id} \times \eta) \circ \langle \pi_1, \mathrm{id}\rangle \diamond \tau_{A,B}\langle h, g\rangle) \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ (\eta \circ \langle \pi_1, \mathrm{id}\rangle \diamond \tau_{A,B}\langle h, g\rangle) && \text{[by 1.7]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ \mu \circ T\eta \circ T\langle \pi_1, \mathrm{id}\rangle \circ \tau_{A,B}\langle h, g\rangle && \text{[by 1.5, 1.3]} \\
&\quad= \mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ T\langle \pi_1, \mathrm{id}\rangle \circ \tau_{A,B}\langle h, g\rangle && \text{[by Def. 1.7]}
\end{aligned}$$

$$\begin{aligned}&= \mu \circ T\tau_{A,C} \circ T\langle \pi_1, f\rangle \circ \tau_{A,B}\langle h, g\rangle \\ &= \tau_{A,C}\langle \pi_1, f\rangle \diamond \tau_{A,B}\langle h, g\rangle. && \text{[by 1.5, 1.3]}\end{aligned}$$

Equation (1.9). By (1.5) and (1.3): $\tau_{A,C}(\mathrm{id} \times f \diamond g) = \tau_{A,C} \circ (\mathrm{id} \times \mu) \circ (\mathrm{id} \times Tf) \circ (\mathrm{id} \times g)$. By the right bottom diagram from the definition of strength this is equal to $\mu \circ T\tau_{A,C} \circ \tau_{A,TC} \circ (\mathrm{id} \times Tf) \circ (\mathrm{id} \times g)$. By naturality of strength, the latter is equal to $\mu \circ T\tau_{A,C} \circ T(\mathrm{id} \times f) \circ \tau_{A,B}(\mathrm{id} \times g) = \mu \circ T(\tau_{A,C}(\mathrm{id} \times f)) \circ \tau_{A,B}(\mathrm{id} \times g)$ and we are done by (1.5), (1.3). □

In terms of effectful programming, the existence of strength opens an opportunity to interleave effectful computations with trivial computations (the ones doing nothing, but pushing forward the arguments), or in other words to capture programming with more than one variable. In fact, the language of strong monads, which we further call the *metalanguage of effects (ME)*, is the simplest language of effectful computation. It provides a uniform basis for all the other more advanced languages. ME extends multisorted logic, defined in Section 1.2 by introducing a unary type constructor T, e.g. by extending the type system to

$$\mathrm{Type}^T_W ::= \mathcal{W} \mid 1 \mid \mathrm{Type}^T_W \times \mathrm{Type}^T_W \mid T(\mathrm{Type}^T_W)$$

and completing the language by two new ingredients: *return* and *binding* as follows.

$$\frac{\Gamma \rhd t : A}{\Gamma \rhd \mathrm{ret}\, t : TA} \qquad\qquad \frac{\Gamma \rhd p : TA \quad \Gamma, x : A \rhd q : TB}{\Gamma \rhd \mathrm{do}\ x \leftarrow p; q : TB}$$

The semantics of the type constructor T is provided by the definition $[\![TA]\!] = T[\![A]\!]$. The semantics of terms is extended by two more rules:

$$\frac{[\![\Gamma \rhd t : A]\!] = g}{[\![\Gamma \rhd \mathrm{ret}\, t : TA]\!] = \eta_{[\![A]\!]} \circ g} \qquad \frac{[\![\Gamma \rhd p : TA]\!] = g \quad [\![\Gamma, x : A \rhd q : TB]\!] = h}{[\![\Gamma \rhd \mathrm{do}\ x \leftarrow p; q : TB]\!] = h \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}_{[\![\Gamma]\!]}, g\rangle}$$

If $(\Gamma \rhd t : A)$ is a well-defined term, we denote by $\mathrm{Vars}(t)$ the set of free variables occurring in t. Like in the case of equational logic, it is easy to see that $\mathrm{Vars}(t) \subseteq \mathrm{Vars}(\Gamma_v)$. Based on this observation one can easily show that given two appropriate terms p, q and a variable x such that $p[q/x]$ is well-defined in sense of equational logic, no free variable occurrence in q becomes bound in $p[q/x]$. As a result, the correct notion of substitution for the terms of ME is the same as for the terms of equational logic: when calculating $p[q/x]$ we do not need to bother about renaming bound variables, we only need to literally replace all the occurrences of x in p by q. The underlying reason for this is because reasoning with term contexts effectively rules out unwanted instances. The reverse of the coin here is that once we strictly adhere to the term-construction rules we cannot create some certainly meaningful terms, e.g. $(\mathrm{do}\ x \leftarrow f(x); g(x))$. Still, we will occasionally use terms like this if they can be reduced to some well-defined ones

$$\textbf{(unit}_1\textbf{)}\quad \mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x = p \qquad \textbf{(unit}_2\textbf{)}\quad \mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q = q[p/x]$$

$$\textbf{(assoc)}\quad \mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow p; q); r = \mathrm{do}\ y \leftarrow p; x \leftarrow q; r \qquad (y \notin FV(r))$$

FIGURE 1.4: Monad laws.

by renaming bound variables by α-conversion. E.g. the latter example corresponds to a well-defined term $(\mathrm{do}\ y \leftarrow f(x); g(y))$. The choice of variable y here is irrelevant (Lemma 1.14).

We shall commonly write $(\mathrm{do}\ x \leftarrow p; y \leftarrow q; r)$ instead of $(\mathrm{do}\ x \leftarrow p; (\mathrm{do}\ y \leftarrow q; r))$. Also we use the notation $(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; q)$ as a shortening of $(\mathrm{do}\ x_1 \leftarrow p_1; \ldots; x_n \leftarrow p_n; q)$ and refer to the fragments $\bar{x} \leftarrow \bar{p}$ as *program sequences*. If a variable x does not appear in a term q then we often shorten $(\mathrm{do}\ x \leftarrow p; q)$ down to $(\mathrm{do}\ p; q)$. Given a program p whose return types is $T(A_1 \times \ldots \times A_n)$ we use $(\mathrm{do}\ \bar{z} \leftarrow p; q)$ as a shortening for

$$\mathrm{do}\ z \leftarrow p; q[\mathrm{pr}^n_1(z)/z_1, \ldots, \mathrm{pr}^n_n(z)/z_n].$$

where z is an appropriate fresh variable whose choice is going to be irrelevant, as further justified by Lemma 1.14. Programs not containing the return operator we call ret-*free*.

The proof calculus ME, corresponding to the metalanguage of effects, extends EQ by three *monad laws* presented in Fig. 1.4 and the evident congruence rules **(cong_bind)** and **(cong_ret)**. The rules **(unit$_1$)**, **(unit$_2$)** and **(assoc)** are called *first unit law*, *second unit law* and *associativity law*, respectively. Terms of the metalanguage of effects as well as of the further extensions of it will also be referred to as *programs*. We call a type $A \in \mathrm{Type}^T_W$ *T-free* if the type constructor T does not occur in A. We call it *computational* if it has form TB for some $B \in \mathrm{Type}^T_W$.

Lemma 1.14. *Given appropriately typed programs p, q and variables x, y such that $x \notin \mathrm{Vars}(q)$:*

$$\vdash_{\mathsf{ME}} \mathrm{do}\ x \leftarrow p; q = \mathrm{do}\ y \leftarrow p; q[y/x],$$

i.e. α-conversion is derivable in ME.

Proof. The proof is given by the calculation:

$$\begin{aligned}
&\mathrm{do}\ x \leftarrow p; q \\
&\quad = \mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow p; \mathrm{ret}\, y); q && [\text{by } \textbf{(unit}_1\textbf{)}] \\
&\quad = \mathrm{do}\ y \leftarrow p; x \leftarrow \mathrm{ret}\, y; q && [\text{by } \textbf{(assoc)}] \\
&\quad = \mathrm{do}\ y \leftarrow p; q[y/x] && [\text{by } \textbf{(unit}_2\textbf{)}]
\end{aligned}$$

□

In spite of the fact that α-conversion is admissible, most of our further results are easier to formulate modulo α-conversion. Therefore we will use α-conversion as the default equivalence relation over programs. In case we need to state syntactic equivalence of two programs, we use the equivalence relation $\doteq$ introduced earlier.

Lemma 1.15 (Rewriting rationale)**.** *Let p, q be two programs with the same return type and let Φ be a set of program equations. Then $\Phi \vdash_{\mathsf{ME}} p = q$ iff there is a sequence of programs $w_1, \ldots, w_n$ such that $p \doteq w_1$, $q \doteq w_n$ and for every $i < n$, w_i and w_{i+1} are presentable in the forms $w_i \doteq C\{u\sigma\}$, $w_{i+1} \doteq C\{r\sigma\}$ for some context C, a variable substitution σ and programs u, r such that either $(u = r) \in \Phi$ or $(r = u) \in \Phi$ or $\vdash_{\mathsf{ME}} r = u$.*

Proof. First of all, we extend the result achieved in Lemma 1.4 to the case of ME, i.e. we prove that $\Phi \vdash_{\mathsf{ME}} p = q$ is equivalent to $\vdash_{\mathsf{ME}_\Phi} p = q$ where by ME_Φ we denote the calculus, obtained from ME by replacing the rule **(inst)** by the rule **(inst$_\Phi$)**. If $\vdash_{\mathsf{ME}_\Phi} p = q$ then the proof of $\Phi \vdash_{\mathsf{ME}} p = q$ is the same as in Lemma 1.4. The converse implication amounts to proving the admissibility of **($\overline{\textbf{inst}}$)** (see the proof of Lemma 1.4) in ME_Φ. This can be shown by induction over the complexity of the proof of the premise of **($\overline{\textbf{inst}}$)**. By virtue of the proof of Lemma 1.4 it suffices to verify the following cases.

The premise is obtained by **(cong_ret)***.* We have thus the following derivation in ME_Φ:

$$\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma, \bar{x} : \bar{A} \rhd s = t : D}{\Gamma, \bar{x} : \bar{A} \rhd \mathsf{ret}\, s = \mathsf{ret}\, t : TD}\,\textbf{(cong\_ret)}}{\Gamma \rhd \mathsf{ret}\, s[\bar{r}/\bar{x}] = \mathsf{ret}\, t[\bar{r}/\bar{x}] : TD}\,(\overline{\textbf{inst}})$$

which is equivalent to

$$\dfrac{\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \Gamma, \bar{x} : \bar{A} \rhd s = t : D}{\Gamma \rhd s[\bar{r}/\bar{x}] = t[\bar{r}/\bar{x}] : D}\,(\overline{\textbf{inst}})}{\Gamma \rhd \mathsf{ret}\, s = \mathsf{ret}\, t : TD}\,\textbf{(cong\_ret)}$$

and thus we can get rid of the application of **($\overline{\textbf{inst}}$)** by induction hypothesis.

The premise is obtained by **(cong_bind)***.* We have the following derivation:

$$\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \dfrac{\Gamma, \bar{x} : \bar{A} \rhd s_1 = s_2 : TA \qquad \Gamma, \bar{x} : \bar{A}, x : A \rhd t_2 = t_2 : TD}{\Gamma, \bar{x} : \bar{A} \rhd \mathsf{do}\ x \leftarrow s_1; t_1 = \mathsf{do}\ x \leftarrow s_2; t_2 : TD}\,\textbf{(bind\_ret)}}{\Gamma \rhd (\mathsf{do}\ x \leftarrow s_1; t_1)[\bar{r}/\bar{x}] = (\mathsf{do}\ x \leftarrow s_2; t_2)[\bar{r}/\bar{x}] : TD}\,(\overline{\textbf{inst}})$$

whose premises are provable in ME_Φ. By induction hypothesis and the following derivations

$$\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \Gamma, \bar{x} : \bar{A} \rhd s_1 = s_2 : TA}{\Gamma \rhd s_1[\bar{r}/\bar{x}] = s_2[\bar{r}/\bar{x}] : TA}\,(\overline{\textbf{inst}})$$

$$\dfrac{\Gamma \rhd r_i : A_i \in \mathcal{T}_\Sigma \qquad \Gamma, \bar{x} : \bar{A}, x : A \rhd t_2 = t_2 : TD}{\Gamma, x : A \rhd t_2[\bar{r}/\bar{x}] = t_2[\bar{r}/\bar{x}] : TD}\,(\overline{\textbf{inst}})$$

both $(\Gamma \rhd s_1[\bar{r}/\bar{x}] = s_2[\bar{r}/\bar{x}] : TA)$ and $(\Gamma, x : A \rhd t_2[\bar{r}/\bar{x}] = t_2[\bar{r}/\bar{x}] : TD)$ must be provable in ME_Φ. Hence $(\Gamma \rhd \mathrm{do}\ x \leftarrow s_1[\bar{r}/\bar{x}]; t_1[\bar{r}/\bar{x}] = \mathrm{do}\ x \leftarrow s_2[\bar{r}/\bar{x}]; t_2[\bar{r}/\bar{x}] : TD)$ is also provable in ME_Φ as justified by the derivation:

$$\frac{\Gamma \rhd s_1[\bar{r}/\bar{x}] = s_2[\bar{r}/\bar{x}] : TA \quad \Gamma, x : A \rhd t_2[\bar{r}/\bar{x}] = t_2[\bar{r}/\bar{x}] : TD}{\Gamma \rhd \mathrm{do}\ x \leftarrow s_1[\bar{r}/\bar{x}]; t_1[\bar{r}/\bar{x}] = \mathrm{do}\ x \leftarrow s_2[\bar{r}/\bar{x}]; t_2[\bar{r}/\bar{x}] : TD}\ \textbf{(bind\_ret)}$$

We are done since $(\mathrm{do}\ x \leftarrow s_k; t_k)[\bar{r}/\bar{x}] \doteq (\mathrm{do}\ x \leftarrow s_k[\bar{r}/\bar{x}]; t_k[\bar{r}/\bar{x}])$ for $k = 1, 2$.

Let us now continue the main proof. Suppose $\Phi \vdash_{\mathsf{ME}} p = q$. As we have seen, this implies $\vdash_{\mathsf{ME}_\Phi} p = q$. We prove the existence of the w_i by induction over the complexity of the latter proof. The induction invariant holds trivially for all the axioms of ME with $C := \Box$ and $\sigma := [\,]$. All the congruence rules, including **(cong_ret)** and **(cong_bind)** easily follow by induction hypothesis: to obtain the goal from the induction hypothesis we only need to redefine the context C. The rules **(sym)** and **(trans)** follow trivially. Finally, consider the rule **(inst$_\Phi$)**. Suppose, $(\bar{x} : \bar{A} \rhd t = s : A) \in \Phi$ and for every i, $(\Gamma \rhd t_i : A_i) \in \mathcal{T}_\Sigma$. By **(inst$_\Phi$)** we conclude the equation $(\Gamma \rhd t\sigma = s\sigma : A)$ under $\sigma := [t_1/x_1, \ldots, t_n/x_n]$ and the evident assignments for the w_i.

Now let us assume that the w_i exist and show that $\Phi \vdash_{\mathsf{ME}} p = q$. By transitivity it suffices to consider the restricted case: $w_1 \doteq p$, $w_2 \doteq q$. By assumption, there exists a context C, a substitution σ and programs u, r such that $w_1 \doteq C\{u\sigma\}$, $w_2 \doteq C\{r\sigma\}$ and either $\vdash_{\mathsf{ME}} u = r$ or $(u = r) \in \Phi$ or $(r = u) \in \Phi$. In all these cases $\Phi \vdash_{\mathsf{ME}} u = r$ from which we conclude $\Phi \vdash_{\mathsf{ME}} p = q$ by **(inst)** and the congruence rules corresponding to the operators from which C is built. □

The following theorem essentially states that the metalanguage of effects is an internal language of strong monads.

Theorem 1.16 (Soundness and completeness [Mog91])**.** *The metalanguage of effects is sound and complete over strong monads.*

Proof. Soundness. By virtue of Theorem 1.6 we only need to verify the monad laws and the new congruence rules. For instance, for **(unit$_2$)** one has

$$\begin{aligned}
&[\![\Gamma \rhd \mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q : TB]\!] \\
&\quad = [\![\Gamma, x : A \rhd q : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}_{[\![\Gamma]\!]}, [\![\Gamma \rhd \mathrm{ret}\, p : TA]\!] \rangle && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma, x : A \rhd q : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}_{[\![\Gamma]\!]}, \eta_{[\![A]\!]} \circ [\![\Gamma \rhd p : A]\!] \rangle && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma, x : A \rhd q : TB]\!] \diamond \eta_{[\![\Gamma]\!] \times [\![A]\!]} \circ \langle \mathrm{id}_{[\![\Gamma]\!]}, [\![\Gamma \rhd p : A]\!] \rangle && \text{[by 1.7]} \\
&\quad = [\![\Gamma, x : A \rhd q : TB]\!] \circ \langle \mathrm{id}_{[\![\Gamma]\!]}, [\![\Gamma \rhd p : A]\!] \rangle && \text{[by 1.2 and 1.5]} \\
&\quad = [\![\Gamma, x : A \rhd q : TB]\!] \circ [\![\Gamma \rhd \langle \Gamma_v, p \rangle : \Gamma_t \times A]\!] && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma \rhd q[p/x] : TB]\!] && \text{[by 1.1]}
\end{aligned}$$

Verification of the other two laws and the congruence rules is analogous.

Completeness. By completing the term category construction of Theorem 1.6: instead of terms of equational logic modulo provable equivalence in EQ we take the terms of ME modulo the provable equivalence in ME. The result is the monad structure $\mathbb{T}_{\Sigma,\Phi}$, over $\mathbf{C}_{\Sigma,\Phi}$, defined as follows. For every object A, the action of the functorial part T of $\mathbb{T}_{\Sigma,\Phi}$ upon the objects of $\mathbf{C}_{\Sigma,\Phi}$ is by syntactic juxtaposition: TA. Monad operators are provided by the assignments:

$$\begin{aligned}
\eta_A &:= [x : A \rhd \mathrm{ret}\, x : TA]_\sim,\\
[x : A \rhd p : TB]^\dagger_\sim &:= [t : TA \rhd \mathrm{do}\ x \leftarrow t; p : TB]_\sim,\\
\tau_{A,B} &:= [t : (A \times TB) \rhd \mathrm{do}\ x \leftarrow \mathrm{snd}(t); \mathrm{ret}\langle \mathrm{fst}(t), x\rangle : T(A \times B)]_\sim.
\end{aligned}$$

Verification of the monad identities runs routinely as expected. □

The metalanguage of effects can be used as a convenient abstraction for effectful imperative programs written in conventional programming languages. Consequently, the calculus ME can be used for proving (equational) properties of such programs.

Example 1.2. A simple illustration of the metalanguage of effects at work is an encoding of program `swap` switching around values of two variables of an imperative programming language. Let the variables be called a and b. Standardly, `swap` is implemented with help of an auxiliary variable, say c. The program then would look as follows.

```
c:=a; a:=b; b:=c
```

Variables `a`, `b` and `c` are *object variables*, in contrast to the *metavariables* which are the variables of ME. For the sake of simplicity we assume that there are no other object variables, nor memory cells to save anything but the values of `a`, `b` and `c`.

Then the program can be presented by the following term.

$$\begin{aligned}
\mathrm{do}\ x &\leftarrow \mathrm{get}_a; \mathrm{put}_c(x);\\
y &\leftarrow \mathrm{get}_b; \mathrm{put}_a(y);\\
z &\leftarrow \mathrm{get}_c; \mathrm{put}_b(z)
\end{aligned}$$

where $\mathrm{get}_t : TA$ and $\mathrm{put}_t : A \to T1$ with $t \in \{a, b, c\}$ implement random access to the memory with the obvious intuitive meaning. However an axiomatisation of them in the presence of generic effects is not immediate. In particular, if we really want to capture typical real-life program behaviour we should agree that both put and get might not terminate. For example, the equality

$$\mathrm{do}\ x \leftarrow \mathrm{get}_a; \mathrm{put}_a(x) = \mathrm{ret}\star$$

might not hold, because there is no evidence that a is initialised at the moment we try to read it, which means that the left-hand side of the latter equality presents a non-terminating program, whereas the right-hand side (a void program, doing nothing) is supposed to always terminate. Furthermore, the equation

$$\mathrm{do}\ x \leftarrow \mathrm{get}_a; \mathrm{put}_a(x); \mathrm{get}_a = \mathrm{get}_a$$

might also not hold under some sensible interpretations, e.g. in case if 'put' and 'get' are object methods of some object-oriented language like Java. It is quite a typical situation that its object fields are allowed to be read but not updated. If that is the case, then the left-hand side would throw an exception but the right-hand side would not, and therefore the equality in question would fail.

This simple analysis clarifies the role of the underlying axiomatisation — the point of the latter is to capture relevant abstract properties of the computational model in order to use them for reasoning about effectful programs. We illustrate in detail how this kind of reasoning is performed in ME. To that end we settle upon the following set of axioms.

(put_get$_\mathbf{t}$) $\quad \mathrm{do}\ \mathrm{put}_t(x); \mathrm{get}_t = \mathrm{do}\ \mathrm{put}_t(x); \mathrm{ret}\, x$

(put_put$_\mathbf{st}$) $\quad \mathrm{do}\ \mathrm{put}_s(x); \mathrm{put}_t(y) = \mathrm{do}\ \mathrm{put}_t(y); \mathrm{put}_s(x)$

(put_put$_\mathbf{t}$) $\quad \mathrm{do}\ \mathrm{put}_t(x); \mathrm{put}_t(y) = \mathrm{put}_t(y)$

where s and t range over $\{a, b, c\}$ and $s \neq t$. This axiomatisation is rather general. In particular, it is compatible with both the identities considered above. A simple, sensible model of this axiomatisation is the partial state monad: $TA = S \to (A \times S) + 1$ over **Set** with $S = (V + 1)^{\{a,b,c\}}$ where V is a set of values (for instance, integers) and the 'get', 'put' operators are defined by the equations:

$$\begin{aligned} \mathrm{get}_t &:= \lambda f.\, \mathrm{case}\ f(t)\ \mathrm{of}\ \mathrm{inl}(x) \to \mathrm{inl}\langle x, f\rangle; \mathrm{inr}(x) \to \mathrm{inr}(\star), \\ \mathrm{put}_t(x) &:= \lambda f.\, \mathrm{inl}\langle f\{t \mapsto x\}, \star\rangle \end{aligned}$$

where $f\{t \mapsto x\}$ is the function, sending s to $f(s)$, for $s \neq t$ and t to x. Sure enough, this interpretation is not unique, but it provides a good intuition about what goes on.

Now suppose we would like to prove that `swap` being called twice in a row brings the memory to the original state. Adapted to our settings, this can be expressed as follows:

$$\mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathtt{swap}; \mathtt{swap} = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_c(y). \tag{1.11}$$

Note that, strictly speaking, the memory state after the double swap is not precisely the same as before, because the auxiliary variable c gets initialised if it was not initialised before, and also gets updated by the value of b. This is exactly captured by (1.11). In order to prove (1.11) the equation

$$\mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathtt{swap} = \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(x); \mathrm{put}_c(x) \tag{1.12}$$

as a lemma. It would imply (1.11) as follows.

$$\begin{aligned}
&\mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathtt{swap}; \mathtt{swap} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(x); \mathrm{put}_c(x); \mathtt{swap} && \text{[by 1.12]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_c(x); \mathrm{put}_b(x); \mathtt{swap} && \text{[by } \textbf{(put\_put}_{\mathbf{bc}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_c(x); \mathrm{put}_a(y); \mathrm{put}_b(x); \mathtt{swap} && \text{[by } \textbf{(put\_put}_{\mathbf{ac}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_c(x); \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_c(y) && \text{[by 1.12]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_c(x); \mathrm{put}_b(y); \mathrm{put}_c(y) && \text{[by } \textbf{(put\_put}_{\mathbf{ca}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_c(x); \mathrm{put}_c(y) && \text{[by } \textbf{(put\_put}_{\mathbf{cb}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_c(y) && \text{[by } \textbf{(put\_put}_{\mathbf{c}}\textbf{)}\text{]}
\end{aligned}$$

The proof of (1.12) is as follows:

$$\begin{aligned}
&\mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathtt{swap} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); x \leftarrow \mathrm{get}_a; \mathrm{put}_c(x); \\
&\qquad\quad y \leftarrow \mathrm{get}_b; \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by def. of } \mathtt{swap}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(x); x \leftarrow \mathrm{get}_a; \mathrm{put}_c(x); \\
&\qquad\quad y \leftarrow \mathrm{get}_b; \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{ab}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(x); x \leftarrow \mathrm{ret}\, x; \mathrm{put}_c(x); \\
&\qquad\quad y \leftarrow \mathrm{get}_b; \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_get}_{\mathbf{a}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(x); \mathrm{put}_c(x); y \leftarrow \mathrm{get}_b; \\
&\qquad\quad \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(unit}_{\mathbf{2}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_c(x); y \leftarrow \mathrm{get}_b; \\
&\qquad\quad \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{ba}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_c(x); \mathrm{put}_b(y); y \leftarrow \mathrm{get}_b; \\
&\qquad\quad \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{bc}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_c(x); \mathrm{put}_b(y); y \leftarrow \mathrm{ret}\, y; \\
&\qquad\quad \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_get}_{\mathbf{b}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_a(x); \mathrm{put}_c(x); \mathrm{put}_b(y); \mathrm{put}_a(y); \\
&\qquad\quad z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(unit}_{\mathbf{2}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_c(x); \mathrm{put}_a(x); \mathrm{put}_b(y); \mathrm{put}_a(y); \\
&\qquad\quad z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{ac}}\textbf{)}\text{]} \\
&\quad = \mathrm{do}\ \mathrm{put}_c(x); \mathrm{put}_b(y); \mathrm{put}_a(x); \mathrm{put}_a(y);
\end{aligned}$$

$$
\begin{aligned}
& z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{ab}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_c(x); \mathrm{put}_b(y); \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{a}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_c(y); \mathrm{put}_a(y); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{cb}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(y); \mathrm{put}_c(x); z \leftarrow \mathrm{get}_c; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_put}_{\mathbf{ca}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(y); \mathrm{put}_c(x); z \leftarrow \mathrm{ret}\, x; \mathrm{put}_b(z) && \text{[by } \textbf{(put\_get}_{\mathbf{c}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_b(y); \mathrm{put}_a(y); \mathrm{put}_c(x); \mathrm{put}_b(x) && \text{[by } \textbf{(unit}_{\mathbf{2}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(y); \mathrm{put}_c(x); \mathrm{put}_b(x) && \text{[by } \textbf{(put\_put}_{\mathbf{ba}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(y); \mathrm{put}_b(x); \mathrm{put}_c(x) && \text{[by } \textbf{(put\_put}_{\mathbf{cb}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(y); \mathrm{put}_b(x); \mathrm{put}_c(x) && \text{[by } \textbf{(put\_put}_{\mathbf{b}}\textbf{)}] \\
&= \mathrm{do}\ \mathrm{put}_a(y); \mathrm{put}_b(x); \mathrm{put}_c(x) && \text{[by } \textbf{(put\_put}_{\mathbf{b}}\textbf{)}]
\end{aligned}
$$

Terms of the metalanguage of effects can be normalised analogously to the terms of simple-typed λ-calculus with injective pairing and terminal objects [Cd96]. We adopt the following definition.

Definition 1.17 (Generalised unit types)**.** Let $\mathcal{U}$ be the smallest set, containing the unit type and closed under products. We call the elements of $\mathcal{U}$ *generalised unit types*. The *canonical element* e_E of a generalised unit type E is $\star$ if $E = 1$ and $\langle e_{E_1}, e_{E_2} \rangle$ if $E = E_1 \times E_2$. For every program t let $\mathsf{nf}_\star(t)$ be the program, obtained from t by replacing every subterm s with return type $E \in \mathcal{U}$ by e_E. A program t is $\star$*-normal* if $t = \mathsf{nf}_\star(t)$.

Alternatively, one can consider $\mathsf{nf}_\star(t)$ as the normal form of t under the reduction rule:

$\star$-rule: $$(p : E) \rightarrowtail e_E \qquad (p \neq e_E)$$

Now we are ready to present rewrite rules which, together with the $\star$-rule, make up a complete rewriting system, capturing all the provable identities of ME. It is useful to adhere to the standard partition into β-like rules and η-like rules.

β-rules:

$$
\begin{gathered}
\mathrm{fst}\langle p, q\rangle \rightarrowtail p \qquad \mathrm{snd}\langle p, q\rangle \rightarrowtail q \\
\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q \rightarrowtail q[p/x] \\
\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow p; q); r \rightarrowtail \mathrm{do}\ y \leftarrow p; x \leftarrow q; r \qquad (y \notin FV(r))
\end{gathered}
$$

η-rules:

$$
\begin{gathered}
\langle \mathrm{fst}(p), e_E\rangle \rightarrowtail p \qquad \langle e_E, \mathrm{snd}(p)\rangle \rightarrowtail p \\
\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, e_E \rightarrowtail p \qquad \mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x \rightarrowtail p \\
\langle \mathrm{fst}(p), \mathrm{snd}(p)\rangle \rightarrowtail p
\end{gathered}
$$

The term contexts here were dropped as usual and can be easily read from the terms, except possibly the two topmost η-rules. Their uncut versions look as follows.

$$\Gamma \rhd \langle \mathsf{fst}(p), e_E \rangle : A \times E \rightarrowtail \Gamma \rhd p : A \times E$$
$$\Gamma \rhd \langle e_E, \mathsf{snd}(p) \rangle : E \times A \rightarrowtail \Gamma \rhd p : E \times A$$

A few comments are in order about how precisely term rewriting is implemented. In contrast to the first-order case we do not need to instantiate the rules by applying variable substitutions because the letters p, q, r, occurring in them are already supposed to run over all terms. Template variables of this kind will occasionally be referred to as *metavariables*. Therefore, by definition, t rewrites to s by a rule $p \rightarrowtail q$ if $t \doteq C\{p\}$ and $s \doteq C\{q\}$ for some reduction context C containing the hole. We refer to p as *redex* and to q as *contractum*.

The terms 'β-reduction', 'β-normalisation', 'β-normal form', etc., shall obviously refer to rewriting under β-rules, and the same convention applies to η-, $\star$-, etc. We denote the β-reduction relation by $\rightarrowtail_\beta$ and the η-reduction relation by $\rightarrowtail_\eta$. It is straightforward to see that $\beta\eta$-reductions always transform $\star$-normal programs to $\star$-normal ones.

$\beta\eta$-rewriting turns out to be confluent and strongly normalising. Before proving this formally let us summarise a few facts from the theory of abstract reduction systems.

Definition 1.18 (Commutation, quasi-commutation [BD86])**.** Let $\rightarrowtail_{\mathsf{a}}$ and $\rightarrowtail_{\mathsf{b}}$ be two reduction relations and let $\rightarrowtail_{\mathsf{ab}} := \rightarrowtail_{\mathsf{a}} \cup \rightarrowtail_{\mathsf{b}}$. Then

– $\rightarrowtail_{\mathsf{b}}$ is said to *commute* over $\rightarrowtail_{\mathsf{a}}$ if $s \rightarrowtail_{\mathsf{a}} . \rightarrowtail_{\mathsf{b}} t$ implies $s \rightarrowtail_{\mathsf{b}} . \rightarrowtail_{\mathsf{a}} t$,

– $\rightarrowtail_{\mathsf{b}}$ is said to *quasi-commute* over $\rightarrowtail_{\mathsf{a}}$ if $s \rightarrowtail_{\mathsf{a}} . \rightarrowtail_{\mathsf{b}} t$ implies $s \rightarrowtail_{\mathsf{b}} . \rightarrowtail_{\mathsf{ab}}^\star t$.

Lemma 1.19. [BD86] *Let $\rightarrowtail_{\mathsf{a}}$ and $\rightarrowtail_{\mathsf{b}}$ be two reduction relations, and let $\rightarrowtail_{\mathsf{b}}$ quasi-commute over $\rightarrowtail_{\mathsf{a}}$. Then $\rightarrowtail_{\mathsf{ab}}$ is strongly normalising iff both $\rightarrowtail_{\mathsf{a}}$ and $\rightarrowtail_{\mathsf{b}}$ are.*

We summarise elementary properties of quasi-commuting relations in the following lemma.

Lemma 1.20. *Let $\rightarrowtail_{\mathsf{a}}$, $\rightarrowtail_{\mathsf{b}}$ and $\rightarrowtail_{\mathsf{c}}$ be three reduction relations, and let $\rightarrowtail_{\mathsf{ab}} := \rightarrowtail_{\mathsf{a}} \cup \rightarrowtail_{\mathsf{b}}$.*

1. *Every reduction relation quasi-commutes over itself.*

2. *If $\rightarrowtail_{\mathsf{c}}$ quasi-commutes over both $\rightarrowtail_{\mathsf{a}}$ and $\rightarrowtail_{\mathsf{b}}$ then it quasi-commutes over $\rightarrowtail_{\mathsf{ab}}$.*

3. *If $\rightarrowtail_{\mathsf{b}}$ quasi-commutes over $\rightarrowtail_{\mathsf{a}}$ then it quasi-commutes over $\rightarrowtail_{\mathsf{a}}^\star$.*

Corollary 1.21. *Given two reduction relations $\rightarrowtail_{\mathsf{a}}$ and $\rightarrowtail_{\mathsf{b}}$, if $\rightarrowtail_{\mathsf{b}}$ quasi-commutes over $\rightarrowtail_{\mathsf{a}}$ then $\rightarrowtail_{\mathsf{b}}$ commutes over $\rightarrowtail_{\mathsf{ab}}^\star$ where $\rightarrowtail_{\mathsf{ab}} := \rightarrowtail_{\mathsf{a}} \cup \rightarrowtail_{\mathsf{b}}$.*

Lemma 1.22. *Let $\rightarrowtail_{\mathrm{a}}$ and $\rightarrowtail_{\mathrm{b}}$ be two reduction relations such that $\rightarrowtail_{\mathrm{b}}$ is strongly normalising and quasi-commutes over $\rightarrowtail_{\mathrm{a}}$. Then $\rightarrowtail_{\mathrm{b}}^{+}$ commutes over $\rightarrowtail_{\mathrm{a}}^{\star}$.*

Proof. It is proven in [Str06] that under the assumptions made, $\rightarrowtail_{\mathrm{b}}$ *semi-commutes over* $\rightarrowtail_{\mathrm{a}}$, i.e. for every s, t, $s \rightarrowtail_{\mathrm{a}}^{\star} . \rightarrowtail_{\mathrm{b}} t$ implies $s \rightarrowtail_{\mathrm{b}}^{+} . \rightarrowtail_{\mathrm{a}}^{\star} t$.

Suppose we have a reduction chain $s \rightarrowtail_{\mathrm{a}}^{\star} t_1 \rightarrowtail_{\mathrm{b}} \ldots \rightarrowtail_{\mathrm{b}} t_n$ with $n > 0$. In order to prove the claim we need to show that $s \rightarrowtail_{\mathrm{b}}^{+} . \rightarrowtail_{\mathrm{a}}^{\star} t$. Let us show it by induction over n. Then in the case of $n = 1$ we are done by semi-commutativity. Suppose $n > 1$. Then by induction hypothesis $s \rightarrowtail_{\mathrm{b}}^{+} . \rightarrowtail_{\mathrm{a}}^{\star} t_{n-1} \rightarrowtail_{\mathrm{b}} t_n$. By the semi-commutation property there exists a reduction of the form $s \rightarrowtail_{\mathrm{b}}^{+} . \rightarrowtail_{\mathrm{b}}^{+} . \rightarrowtail_{\mathrm{a}}^{\star} t_n$ and thus the proof is completed. □

Observe that $\beta\eta$-rules do not spoil $\star$-normality. This justifies the statement of the following lemma.

Lemma 1.23. *The system of $\beta\eta$-reductions is confluent and strongly normalising over $\star$-normal programs.*

Proof. Strong normalisation. Strong normalisation of $\rightarrowtail_{\beta}$ was proven in [BBdP98]. The relation $\rightarrowtail_{\eta}$ is obviously strongly normalising, because it reduces the term size. In order to prove strong normalisation of the combined relation let us prove that as long as $\star$-normal programs are concerned, $\rightarrowtail_{\beta}$ quasi-commutes over $\rightarrowtail_{\eta}$. By Lemma 1.19 this will imply strong normalisation of $\rightarrowtail_{\beta\eta}$. For instance, let us prove the case when the first reduction step $s \rightarrowtail_{\eta} r$ is made by the first unit law and the second one $r \rightarrowtail_{\beta} t$ by the second unit law respectively. Let $s \doteq C\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\}$ and $r \doteq C\{p\}$. In order to define the reduction $r \rightarrowtail_{\beta} . \rightarrowtail_{\beta\eta}^{\star} t$ we need to range over the following three cases:

1. *The redex is a subterm of p.* Then $t \doteq C\{u\}$ where $p \rightarrowtail_{\beta} u$ and the reduction in question is $s \doteq C\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\} \rightarrowtail_{\beta} C\{\mathrm{do}\ x \leftarrow u; \mathrm{ret}\, x\} \rightarrowtail_{\eta} C\{u\}$.

2. *The redex is a (proper) superterm of p.* Therefore either $C \doteq K\{\mathrm{do}\ y \leftarrow \mathrm{ret}\, M; u\}$ or $C \doteq K\{\mathrm{do}\ y \leftarrow \mathrm{ret}\, u; M\}$ or $p \doteq \mathrm{ret}\, q$ and $C \doteq K\{\mathrm{do}\ y \leftarrow \Box; u\}$. In the first subcase the reduction in question is:

$$\begin{aligned} s &\doteq K\{\mathrm{do}\ y \leftarrow \mathrm{ret}\, M\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\}; u\} \\ &\rightarrowtail_{\beta} K\{u[M\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\}/y]\} \\ &\rightarrowtail_{\eta} K\{u[M\{p\}/y]\}. \end{aligned}$$

   In the second subcase we have:

$$\begin{aligned} s &\doteq K\{\mathrm{do}\ y \leftarrow \mathrm{ret}\, u; M\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\}\} \\ &\rightarrowtail_{\beta} K\{M\{\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x\}[u/y]\} \\ &\doteq K\{M[u/y]\{\mathrm{do}\ x \leftarrow p[u/y]; \mathrm{ret}\, x\}\} \end{aligned}$$

$$\begin{aligned}
&\rightarrowtail_{\eta} K\{M[u/y]\{p[u/y]\}\}\\
&\doteq\ K\{M\{p\}[u/y]\}.
\end{aligned}$$

Finally, in the third subcase we have

$$\begin{aligned}
s &\doteq K\{\mathrm{do}\ y \leftarrow (\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x); u\}\\
&\doteq K\{\mathrm{do}\ y \leftarrow (\mathrm{do}\ x \leftarrow \mathrm{ret}\, q; \mathrm{ret}\, x); u\}\\
&\rightarrowtail_{\beta} K\{\mathrm{do}\ x \leftarrow \mathrm{ret}\, q; y \leftarrow \mathrm{ret}\, x; u\}\\
&\rightarrowtail_{\beta} K\{\mathrm{do}\ y \leftarrow \mathrm{ret}\, q; u\}\\
&\rightarrowtail_{\beta} K\{u[q/y]\}.
\end{aligned}$$

3. *The redex is parallel with p.* This case is trivial because the reductions are completely independent and therefore can be harmlessly rearranged.

The remaining rule combinations can be proven in similar fashion, except for the remarkable example of the following kind, calling for $\star$-normality:

$$\begin{aligned}
&\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow (\mathrm{do}\ z \leftarrow p; \mathrm{ret}\, e_E); q); r\\
\rightarrowtail_{\eta}\ &\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow p; q); r\\
\rightarrowtail_{\beta}\ &\mathrm{do}\ y \leftarrow p; x \leftarrow q; r
\end{aligned}$$

In order to obtain the alternative reduction in question we reduce the original term to $(\mathrm{do}\ x \leftarrow (\mathrm{do}\ z \leftarrow p; y \leftarrow \mathrm{ret}\, e_E; q); r)$ by $\rightarrowtail_{\beta}$. Since y is of type E, q can not depend on y because of $\star$-normality of q. Hence, by applying the second unit law, we obtain $(\mathrm{do}\ x \leftarrow (\mathrm{do}\ z \leftarrow p; q); r)$ which, in turn, can be evidently reduced to the target.

Confluence. The argument standardly appeals to Newman's Lemma (cf. e.g. [Klo92]). According to it, one only needs to prove *local confluence*, i.e. that for any s and t, the span $s \leftarrowtail . \rightarrowtail t$ is joinable; in other words, there is r such that $s \rightarrowtail^{\star} r\ {}^{\star}\!\!\leftarrowtail t$. In the first-order case it is usually done by considering all possible critical pairs. An analogous approach works for the higher-order rewriting [MN98] but it can not be easily adapted to the situation at hand mainly because ME does not support explicit function abstraction. Therefore we need to stick to the general definition of local confluence and analyse all the possible $C_9^2 = 72$ rule combinations.

Observe that if the redexes of the reductions of the span $. \leftarrowtail\ t\ \rightarrowtail .$ occur parallel to each other, i.e. neither of the redexes is a subterm of the other one, the outcomes are trivially joinable. Otherwise, there must exist two contexts C and K and a term t such that $t \doteq C\{K\{s\}\}$ and s, $K\{s\}$ are the redexes. It obviously suffices to prove that s and $K\{s\}$ are joinable. As an example, consider the combination of the second unit law and the associativity law. Let $s \doteq (\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q)$ be the first redex, contracted to $q[p/x]$. If the associativity rule applies in such a way that one of the metavariables matches a superterm of s then the solution is obvious. A more interesting situation arises if one of the metavariables matches some proper subterm of s. This is only possible if $K \doteq (\mathsf{do}\ y \leftarrow \square; r)$, in which case:

$$K\{s\} \doteq \mathsf{do}\ y \leftarrow (\mathsf{do}\ x \leftarrow \mathsf{ret}\, p; q); r \rightarrowtail_{\beta} \mathsf{do}\ x \leftarrow \mathsf{ret}\, p; y \leftarrow q; r.$$

The latter term of this sequence is apparently joinable with $(\mathsf{do}\ y \leftarrow q[p/x]; r)$. The remaining rule combinations can be handled in the same manner. □

Now we can prove that the $\star$-rule can be joined with the $\beta\eta$-rules without spoiling the properties proved in Lemma 1.23.

Theorem 1.24. *The $\beta\eta\star$-reduction is confluent and strongly normalising.*

Proof. Strong normalisation. By contradiction. Suppose there is an infinite chain:

$$t_1 \rightarrowtail_{\beta\eta\star} t_2 \rightarrowtail_{\beta\eta\star} \dots \tag{1.13}$$

It is easily verified that

$$t \rightarrowtail_{\beta\eta\star} s \quad \text{implies} \quad \mathsf{nf}_\star(t) \rightarrowtail^\star_{\beta\eta} \mathsf{nf}_\star(s). \tag{1.14}$$

Hence, by (1.13), there is an infinite reduction sequence of $\star$-normal programs:

$$\mathsf{nf}_\star(t_1) \rightarrowtail^\star_{\beta\eta} \mathsf{nf}_\star(t_2) \rightarrowtail^\star_{\beta\eta} \dots$$

This immediately leads to the contradiction with Lemma 1.23 unless the latter sequence possesses only a finite number of transitions by $\rightarrowtail_{\beta\eta}$, i.e. from some position onwards, $\mathsf{nf}_\star(t_{i+1}) = \mathsf{nf}_\star(t_i)$. Careful examination of the $\beta\eta$-rules proves that $\mathsf{nf}_\star(t_{i+1}) = \mathsf{nf}_\star(t_i)$ only if $t_i \rightarrowtail t_{i+1}$ by a size-decreasing rule. Hence, (1.13) cannot be infinite, contradiction.

Confluence. The proof of confluence established in Lemma 1.23 fully applies here. The only case not covered there is the span of the form $t\ {}_{\beta\eta\star}\!\leftarrowtail u \rightarrowtail_\star s$. According to (1.14), $\mathsf{nf}_\star(u) \rightarrowtail^\star_{\beta\eta} \mathsf{nf}_\star(t)$. Hence t and s can be joined:

$$t \rightarrowtail^\star_\star \mathsf{nf}_\star(t)\ {}^{\star}_{\beta\eta}\!\leftarrowtail \mathsf{nf}_\star(u) = \mathsf{nf}_\star(s)\ {}_\star\!\leftarrowtail s.$$

The proof of the theorem is thus completed. □

Theorem 1.24 points out an evident algorithm for deciding equality of ME programs.

Corollary 1.25. *Equality of ME programs is effectively decidable: two programs are provably equal iff their $\beta\eta\star$-normal forms coincide (up to α-equivalence).*

The established decidability result about unconditional program equality contrasts with the problem of program equality modulo a set of equational axioms.

Theorem 1.26. *There exists a finite set of program identities Φ such that the conditional word problem with respect to Φ is undecidable.*

Proof. Already the word problem in EQ is undecidable because it possesses encoding of the word problem for semigroups which is undecidable as exemplified by the undecidability of the word problem for finitely presented semigroups [Mat67]. □

1.4 Underlying theory for data

The negative result of Theorem 1.26 can be interpreted in terms of the usual programming practice as follows. Signature symbols occurring in p, q can have arguments of non-T-free types. If one is allowed to use any identities whatsoever as the premises Φ, a straightforward idea is to interpret these symbols as new control operators by putting a suitable axiomatisation into Φ. It is pretty clear that in general the possibility of adding such user-defined control operators substantially increases expressivity, and thus the negative result of Theorem 1.26 is no wonder. Using this view of the problem, we may try to circumvent the undecidability issue by totally prohibiting user-defined control operators. The most immediate way to do so is by a simple syntactic restriction.

Definition 1.27 (Plain signatures)**.** We call a signature Σ *plain* if for any $f : A \to B \in \Sigma$ the type A is T-free.

Definition 1.28 (Atomic programs)**.** We call a program p *atomic* if it contains only signature symbols, variables and cartesian primitives. An atomic program is *cartesian* if it does not contain signature symbols (hence built entirely of cartesian primitives). We denote by $\mathcal{A}_\Sigma$ the set of all atomic programs over signature Σ.

Remark 1.29. $\beta\eta\star$-normal programs over plain signatures have a particularly simple form. Every such program is either a cartesian pair of two $\beta\eta\star$-normal programs or has the form $(\mathrm{do}\ \bar{x} \leftarrow \bar{a}; a)$ with a_i, a being $\beta\eta\star$-normal atomic, or has the form $(\mathrm{do}\ \bar{x} \leftarrow \bar{a}; \mathrm{ret}\ p)$ with the a_i being $\beta\eta\star$-normal atomic and p being $\beta\eta\star$-normal.

Lemma 1.30. *Let the underlying signature Σ be plain and let t be a program.*

1. *If $f(t)$ is $\beta\eta\star$-normal and $f \in \Sigma$ then t is atomic.*
2. *If $h(t)$ is $\beta\eta\star$-normal and $h \in \{\mathrm{fst}, \mathrm{snd}\}$ then t is atomic.*
3. *If the return type of t is T-free and t is $\beta\eta\star$-normal then t is atomic.*

Proof. First, observe that (3) implies (1) because by assumption Σ is plain, and thus $f(t)$ types only if the return type of t is T-free. Let us establish (2) and (3) simultaneously by induction over the term complexity of t. Let us consider the possible cases.

– If t is either a variable or $\star$ then we are done trivially.

– If $t \doteq \mathsf{fst}(s)$ or $t \doteq \mathsf{snd}(s)$ with some s then by induction, s is atomic. Therefore t is also atomic.

– Let $t \doteq \langle s, r\rangle$ with some s, r. We only need to prove (3) because by $\beta\eta\star$-normality the present clause is not applicable to (2). Suppose the return type of t is T-free. The return types of s and r are simpler than the return type of t. Therefore they are also T-free, and thus we are done by induction hypothesis.

– If $t \doteq f(s)$ with some s and $f \in \Sigma$ then the return type of s is T-free (because Σ is plain). Therefore we are done by induction hypothesis. □

As we can see, the notion of plain signature indeed allows effective separation of the layer of control operators from the layer of signature symbols. Remark 1.29 essentially points out that w.l.o.g. the control operators are floating on top, strictly over the constructions involving the signature symbols. It is natural to interpret this bottom layer as the layer of data. We justify this in the following definition.

Definition 1.31 (Data theory)**.** An *(equational) data theory* is a set of equations of the form $a = b$ where $a \in \mathcal{A}_\Sigma, b \in \mathcal{A}_\Sigma$ and all the variables in $\mathrm{Vars}(a), \mathrm{Vars}(b)$ are of T-free type.

Example 1.3 (Internal truth values)**.** Suppose, there is a basic sort $\Omega \in \mathcal{W}$ and operations $\top, \bot : 1 \to \Omega, \wedge, \vee, \Rightarrow : \Omega \times \Omega \to \Omega$. An equational theory of Heyting algebras is given by the axioms:

$$a \wedge b = b \wedge a \qquad (a \wedge b) \wedge c = a \wedge (b \wedge c) \qquad a \wedge a = a \qquad a \wedge (a \vee b) = a$$
$$a \vee b = b \vee a \qquad (a \vee b) \vee c = a \vee (b \vee c) \qquad a \vee a = a \qquad a \vee (a \wedge b) = a$$
$$a \wedge (a \Rightarrow b) = a \wedge b \qquad b \wedge (a \Rightarrow b) = b \qquad a \Rightarrow (b \wedge c) = (a \Rightarrow b) \wedge (a \Rightarrow c)$$
$$a \Rightarrow a = \top \qquad \bot \vee a = a \qquad \top \wedge a = a$$

By adding the excluded middle law

$$(p \Rightarrow \bot) \vee p = \top$$

one obtains an equational theory of Boolean algebras.

Example 1.4 (Internal equality)**.** Let $\Omega \in \mathcal{W}$ be an internal truth value type and suppose, in addition, that for every T-free type A there is a binary symbol $eq_A : A \times A \to \Omega$ in the signature. Then for every A the axioms:

$$eq_A(a, a) = \top, \qquad (eq_A(a, b) \Rightarrow eq_A(a, b)) = \top,$$

$$(eq_A(a,b) \wedge eq_A(b,c) \Rightarrow eq_A(a,c)) = \top$$

define an internal equality operator eq_A. Normally, one is interested in equality relations that respect the cartesian structure and are congruences, i.e.

$$(eq_C(a,b) \Rightarrow eq_D(f(a),f(b))) = \top, \qquad eq_\star(c,d) = \top,$$
$$(eq_{A\times B}(\langle a_1,b_1\rangle,\langle a_2,b_2\rangle) \iff eq_{A\times B}(a_1,a_2) \wedge eq_{A\times B}(b_1,b_2)) = \top$$

for every T-free A,B,C,D, every appropriately typed a,b,a_1,a_2,b_1,b_2,c,d and $f \in \Sigma$.

Let us denote by Π the set of variable substitutions $[\bar{t}/\bar{x}]$ such that every t_i is of the form $h_1(\ldots(h_n(v))\ldots)$ where for every k, $h_k \in \{\mathrm{fst},\mathrm{snd}\}$, v is a variable of non-T-free type and $n > 0$. Given a data theory $\mathcal{E}$, we define a relation $\approx_\mathcal{E}$ over programs by the equivalence: for every p,q, $p \approx_\mathcal{E} q$ iff there exists a context C, a substitution $\sigma \in \Pi$ and two atomic programs a,b such that all the variables from $\mathrm{Vars}(p) \cup \mathrm{Vars}(q)$ are T-free, $p \doteq C\{a\sigma\}$, $q \doteq C\{b\sigma\}$ and $\mathcal{E} \vdash_{\mathsf{EQ}} a = b$. By definition, $\approx_\mathcal{E}$ is reflexive and symmetric. Let us denote by $\approx^*_\mathcal{E}$ the transitive closure of $\approx_\mathcal{E}$ which is hence an equivalence. For every natural n we denote by $\approx^n_\mathcal{E}$ the n-th power of $\approx_\mathcal{E}$ with respect to binary relation composition. By definition, for every p,q, $p \approx^*_\mathcal{E} q$ iff $p \approx^n_\mathcal{E} q$ for some n. Given two contexts C and D, $C \approx_\mathcal{E} D$, $C \approx^n_\mathcal{E} D$ and $C \approx^*_\mathcal{E} D$ encode correspondingly: $\forall v.\, C\{v\} \approx_\mathcal{E} D\{v\}$, $\forall v.\, C\{v\} \approx^n_\mathcal{E} D\{v\}$ and $\forall v.\, C\{v\} \approx^*_\mathcal{E} D\{v\}$.

Lemma 1.32. *Suppose we are given a data theory $\mathcal{E}$ over a plain signature Σ, programs p,q,r,u over Σ and a variable x such that $p \approx^*_\mathcal{E} q$, $r \approx^*_\mathcal{E} u$ and the expressions $p[u/x]$, $q[r/x]$ are well-defined. Then there exist s and t such that $p[u/x] \rightarrowtail^\star_{\beta\eta\star} s$, $q[r/x] \rightarrowtail^\star_{\beta\eta\star} t$ and $s \approx^*_\mathcal{E} t$.*

Proof. We prove the claim by induction over the complexity of A, the type of x. Let us consider the possible cases.

- Let A be T-free. Let us first consider the simpler case: $p \approx_\mathcal{E} q$, i.e. for some atomic a,b, a context C and a substitution $\sigma \in \Pi$, $p \doteq C\{a\sigma\}$, $q \doteq C\{b\sigma\}$ and $\mathcal{E} \vdash_{\mathsf{EQ}} a = b$. By definition, the types of all variables which occur in the codomain of σ are necessarily non-T-free, whereas A is T-free. Therefore, $p[u/x] \doteq C[u/x]\{(a[u/x])\sigma\}$, $q[r/x] \doteq C[r/x]\{(b[r/x])\sigma\}$. Observe that both $a[u/x]$, $b[r/x]$ are atomic and $\mathcal{E} \vdash_{\mathsf{EQ}} a[u/x] = b[r/x]$. Therefore, by definition, $p[u/x] \approx^*_\mathcal{E} q[r/x]$. The general case, i.e. if $p \approx^n_\mathcal{E} q$ with $n \neq 1$ now follows easily by induction over n.
- Let $A = A_1 \times A_2$ and A is non-T-free. Let x_1, x_2 be two fresh variables of types A_1 and A_2 correspondingly. Let p' be the program obtained from p by replacing every subterm $\mathrm{fst}(x)$ by x_1, every subterm $\mathrm{snd}(x)$ by x_2 and let q' be the program obtained from q in the same way. Then evidently:

$$p[u/x] \;\doteq\; p'[u/x][\mathrm{fst}(u)/x_1][\mathrm{snd}(u)/x_2],$$

$$q[r/x] \;\doteq\; q'[r/x][\mathrm{fst}(r)/x_1][\mathrm{snd}(r)/x_2].$$

Let us show that $p'[u/x] \approx^*_{\mathcal{E}} q'[r/x]$. First consider the case: $p \approx_{\mathcal{E}} q$, i.e. for some $C, \sigma \in \Pi$ and appropriate atomic a, b, $p \doteq C\{a\sigma\}$, $q \doteq C\{b\sigma\}$ and $\mathcal{E} \vdash_{\mathsf{EQ}} a = b$. Observe that since x is of non-T-free type, neither a nor b can contain it. Let C' be the context, obtained from C in the same way as p' was obtained from p, and let ς be the substitution obtained from σ by applying the analogous transformation to the programs in the codomain. Then $p' \doteq C'\{a\varsigma\}$, $q' \doteq C'\{b\varsigma\}$ and therefore $p'[u/x] \doteq C'[u/x]\{a\varsigma\}$, $q'[r/x] \doteq C'[r/x]\{b\varsigma\}$. Observe that ς is not necessarily an element of Π, because it may map some variables to variables. It is straightforward though to decompose it to $\varsigma_1\varsigma_2$ where ς_1 maps variables to variables and $\varsigma_2 \in \Pi$. Then evidently: $p'[u/x] \doteq C'[u/x]\{(a\varsigma_1)\varsigma_2\}$, $q'[r/x] \doteq C'[r/x]\{(b\varsigma_1)\varsigma_2\}$ and $\mathcal{E} \vdash_{\mathsf{EQ}} a\varsigma_1 = b\varsigma_1$, i.e. $p'[u/x] \approx q'[r/x]$. In general, if $p \approx^*_{\mathcal{E}} q$ i.e. if $p \approx^n_{\mathcal{E}} q$ with some n then $p'[u/x] \approx^* q'[r/x]$ follows by induction over n.

Since the return type of x_1 is simpler than the return type of x, according to induction hypothesis, there exist s' and t' such that $p'[u/x][\mathrm{fst}(u)/x_1] \rightarrowtail^\star_{\beta\eta\star} s'$, $q'[r/x][\mathrm{fst}(r)/x_1] \rightarrowtail^\star_{\beta\eta\star} t'$ and $s' \approx^*_{\mathcal{E}} t'$. By the same reason, there exist t'' and s'' such that $s'[\mathrm{snd}(u)/x_2] \rightarrowtail^\star_{\beta\eta\star} s$, $t'[\mathrm{snd}(u)/x_2] \rightarrowtail^\star_{\beta\eta\star} t$ and $s \approx^*_{\mathcal{E}} t$. Since

$$\begin{aligned} p[u/x] &\doteq p'[u/x][\mathrm{fst}(u)/x_1][\mathrm{snd}(u)/x_2] &&\rightarrowtail^\star_{\beta\eta\star} s'[\mathrm{snd}(u)/x_2] \rightarrowtail^\star_{\beta\eta\star} s,\\ q[r/x] &\doteq q'[r/x][\mathrm{fst}(r)/x_1][\mathrm{snd}(r)/x_2] &&\rightarrowtail^\star_{\beta\eta\star} t'[\mathrm{snd}(u)/x_2] \rightarrowtail^\star_{\beta\eta\star} t, \end{aligned}$$

the programs s and t are those which make the claim true.

– Let $A = TB$ for some B. Once again, we consider first the simpler case: $p \approx_{\mathcal{E}} q$. Let C, σ, a, b be the corresponding data from the definition of $\approx_{\mathcal{E}}$, in particular: $p \doteq C\{a\sigma\}$, $q \doteq C\{b\sigma\}$ and $\mathcal{E} \vdash_{\mathsf{EQ}} a = b$. Since x is of non-T-free type it cannot occur in a, b. For typing reasons x also cannot occur in any program from the codomain of σ. Therefore, $p[u/x] \doteq C[u/x]\{a\sigma\}$, $q[r/x] \doteq C[r/x]\{b\sigma\}$ and thus, evidently, $p[u/x] \approx^*_{\mathcal{E}} q[r/x]$. The general case, i.e. $p \approx^n_{\mathcal{E}} q$ for some n follows by induction over n.

We have now covered all the possible options. The claim is thus proved. □

Lemma 1.33. *Let Σ be a plain signature and let $\mathcal{E}$ be a data theory over it. Then for all programs p and q over Σ, $\mathcal{E} \vdash_{\mathsf{ME}} p = q$ iff $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$.*

Proof. First observe that the implication: $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q) \implies \mathcal{E} \vdash_{\mathsf{ME}} p = q$ immediately follows from the two evident facts: (i) the equivalence relation $\approx^*_{\mathcal{E}}$ is stronger than provable equality in ME modulo the set of axioms $\mathcal{E}$, and (ii) $\mathsf{nf}_{\beta\eta\star}$ transforms programs equivalently under ME.

In order to prove the remaining implication let us assume that $\mathcal{E} \vdash_{\mathsf{ME}} p = q$. By Lemma 1.15 there exists a sequence of programs $w_1, \ldots, w_n$ such that $w_1 \doteq p$, $w_n \doteq q$ and for every $i < n$ either $w_i \rightarrowtail^{\star}_{\beta\eta\star} w_{i+1}$ or $w_{i+1} \rightarrowtail^{\star}_{\beta\eta\star} w_i$ or for some context C, a variable substitution σ and a pair of terms u, r, $w_i \doteq C\{u\sigma\}$, $w_{i+1} \doteq C\{r\sigma\}$ and $(u = r) \in \widetilde{\mathcal{E}}$ where by $\widetilde{\mathcal{E}}$ we denote the symmetric closure of $\mathcal{E}$. Let us show that in the latter case we always can ensure that $w_i \approx^*_{\mathcal{E}} w_{i+1}$. Let σ' be the substitution, obtained by restricting σ to $\mathrm{Vars}(u) \cup \mathrm{Vars}(r)$ and further $\beta\eta\star$-normalisation of the programs in the codomain. Since $\mathcal{E}$ is a data theory, all the variables from $\mathrm{Vars}(u) \cup \mathrm{Vars}(r)$ are T-free and thus, by Lemma 1.30 all the programs in the codomain of σ' are atomic. Observe that a non-T-free variable v can occur in $u\sigma'$ or $r\sigma'$ only in subterms of the form $h_1(\ldots(h_n(v))\ldots)$ whose return type is T-free where every h_k is from $\{\mathrm{fst}, \mathrm{snd}\}$. Let a and b be the atomic programs, obtained from $u\sigma'$, $r\sigma'$ by replacing every such subterm by a fresh variable and let $\varsigma \in \Pi$ be such that $u\sigma' \doteq a\varsigma$, $r\sigma' \doteq b\varsigma$. Let $w'_i := C\{a\varsigma\}$, $w'_{i+1} := C\{b\varsigma\}$. Evidently, $w_i \rightarrowtail^{\star}_{\beta\eta\star} w'_i$, $w_{i+1} \rightarrowtail^{\star}_{\beta\eta\star} w'_{i+1}$ and thus we are done by replacing the section (w_i, w_{i+1}) of the sequence $w_1, \ldots, w_n$ with $(w_i, w'_i, w'_{i+1}, w_{i+1})$. We have thus proved that $\mathcal{E} \vdash_{\mathsf{ME}} p = q$ amounts to: $p\ (\approx_{\mathcal{E}} \cup \rightarrowtail_{\beta\eta\star} \cup {}_{\beta\eta\star}\!\leftarrowtail)^{\star}\ q$. In other words, there exists n such that

$$w_1 := p \rightharpoondown w_2 \rightharpoondown \ldots \rightharpoondown w_n := q \tag{1.15}$$

where $\rightharpoondown \in \{\approx_{\mathcal{E}}, \rightarrowtail_{\beta\eta\star}, {}_{\beta\eta\star}\!\leftarrowtail\}$. Let us show by induction over n that the equivalence $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$ follows from the following statement:

$$\text{for every } s, t \text{ such that } s \approx^*_{\mathcal{E}} t,\ \mathsf{nf}_{\beta\eta\star}(s) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(t). \tag{$*$}$$

If $n = 0$ then $p = q$ and we are trivially done. Suppose $n > 0$. Then $p \rightharpoondown \ldots \rightharpoondown w_{n-1} \rightharpoondown q$ and by induction hypothesis, $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(w_{n-1})$. Let us proceed by case distinction. If $w_{n-1} \approx_{\mathcal{E}} q$ then by ($*$), $\mathsf{nf}_{\beta\eta\star}(w_n) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$ and thus $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$. If $w_{n-1} \rightarrowtail_{\beta\eta\star} q$ or $w_{n-1}\ {}_{\beta\eta\star}\!\leftarrowtail q$ then $\mathsf{nf}_{\beta\eta\star}(w_n) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$ and again $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$. We will be done once we prove ($*$).

Let us show how to ensure $\beta\eta\star$-normality of s and t in ($*$). Let e.g. s be not $\beta\eta\star$-normal. Depending on the $\beta\eta\star$-redex we define s' and t' such that $s \rightarrowtail^{+}_{\beta\eta\star} s'$, $t \rightarrowtail^{\star}_{\beta\eta\star} t'$ and $s' \approx^*_{\mathcal{E}} t'$. Instead of considering all the possible redexes one by one, we focus only on the representative ones.

– Let $s \doteq C\{\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow u; r); w\}$ where C is some context and $y \notin \mathrm{Vars}(w)$. The assumption $s \approx^*_{\mathcal{E}} t$ means in particular that t can be obtained from s by replacing atomic programs with atomic programs. Therefore, t must be of the form $C'\{\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow u'; r'); w'\}$ where $u' \approx^*_{\mathcal{E}} u$, $r' \approx^*_{\mathcal{E}} r$, $w' \approx^*_{\mathcal{E}} w$ and $C' \approx^*_{\mathcal{E}} C$. Let us put $s' := C\{\mathrm{do}\ y \leftarrow u; x \leftarrow r; w\}$, $t' := C'\{\mathrm{do}\ y \leftarrow u'; x \leftarrow r'; w'\}$. It is clear that s' and t' are indeed as announced.

– Let $s \doteq C\{\mathsf{fst}\langle u, r\rangle\}$. If any of the u, r is not atomic then the routine is the same as in the previous case. Suppose both u and r are atomic. If C does not contain $\Box$ then the problem becomes trivial. Let C contain $\Box$ and let $C = K\{M\}$ be the decomposition of C defined as follows: M is the maximal context containing $\Box$ and such that $M\{u\}$ is atomic. Since by assumption, $s \approx^*_{\mathcal{E}} t$, t must be of the form $M'\{w\}$ where $w \approx^*_{\mathcal{E}} K\{\mathsf{fst}\langle u, r\rangle\}$ and $M \approx^*_{\mathcal{E}} M'$. As a consequence: $M\{K\{\mathsf{fst}\langle u, r\rangle\}\} \approx^*_{\mathcal{E}} M'\{w\}$ and thus we are done by putting $s' := C\{u\}$, $t' := t$.

– Let $s \doteq C\{\mathsf{do}\ x \leftarrow \mathsf{ret}\, u; r\}$. By the same arguments as before t must be of the form $C'\{\mathsf{do}\ x \leftarrow \mathsf{ret}\, u'; r'\}$ where $u \approx^*_{\mathcal{E}} u'$, $r \approx^*_{\mathcal{E}} r'$ and $C \approx^*_{\mathcal{E}} C'$. Then $s \rightarrowtail_{\beta\eta\star} C\{r[u/x]\}$, $s \rightarrowtail^{\star}_{\beta\eta\star} C'\{r'[u'/x]\}$. By Lemma 1.32 there exist s' and t' such that $C\{r[u/x]\} \rightarrowtail^{\star}_{\beta\eta\star} s'$ and $C'\{r'[u'/x]\} \rightarrowtail^{\star}_{\beta\eta\star} t'$ and $s' \approx^*_{\mathcal{E}} t'$, and thus we are done.

We have thus shown how to $\beta\eta\star$-reduce s from ($*$) unless it is already $\beta\eta\star$-normal so that the equivalence $s \approx^*_{\mathcal{E}} t$ is maintained. By repeating the routine sufficiently many times we can ensure $\beta\eta\star$-normality of s. In the same way we can ensure $\beta\eta\star$-normality of t. Eventually we obtain the equivalence $\mathsf{nf}_{\beta\eta\star}(s) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(t)$ which proves ($*$). Therefore we are done with the proof of the lemma. □

Now we can prove the main theorem of this section.

Theorem 1.34. *Let Σ be a plain signature and let $\mathcal{E}$ be a data theory over it. If the conditional word problem for $\mathcal{E}$ is decidable, so is the problem of deciding equality of ME programs modulo $\mathcal{E}$.*

Proof. Let e.g. p and q be two programs over Σ of the same return type. By Lemma 1.33 $\mathcal{E} \vdash_{\mathsf{ME}} p = q$ is decidable iff the equivalence $\mathsf{nf}_{\beta\eta\star}(p) \approx^*_{\mathcal{E}} \mathsf{nf}_{\beta\eta\star}(q)$ is decidable. By Remark 1.29 the latter equivalence holds iff $\mathsf{nf}_{\beta\eta\star}(p)$ and $\mathsf{nf}_{\beta\eta\star}(q)$ have the same form on the level of control operator and the atomic subterms of $\mathsf{nf}_{\beta\eta\star}(p)$ can be related with the atomic subprograms of $\mathsf{nf}_{\beta\eta\star}(q)$ by $\approx^*_{\mathcal{E}}$ in a certain order. Therefore the problem in question is decidable if we can decide any equivalence $a \approx^*_{\mathcal{E}} b$ where both a and b are atomic. It is easy to see by definition that if a, b are atomic then $a \approx^*_{\mathcal{E}} b$ iff $\mathcal{E} \vdash_{\mathsf{EQ}} a = b$. The latter problem is decidable by assumption, and thus we are done. □

1.5 Computational monads collectively

In many cases it suffices to fix a particular monad and not to involve any other monads in the reasoning. Moreover, it might be reasonable to completely withdraw from any explicit use of the monad operations and thus switch from the metalanguage of effects to e.g. computational λ-calculus [Mog91]. At the other extreme one may consider every (small) monad as an object of some appropriate category in order to study abstractly various relations between monads, in particular their compositional properties [HPP03, HLPP07, LG02]. Although the problem of monad composition is not touched upon in this thesis (but remains nonetheless of high importance), some general facts about the collective treatment of monads are worth mentioning.

One of the basic concepts serving the goals of monad composition is the notion of a *monad transformer* (implicitly) introduced in [Mog89a] and coined in [LHJ95]. A monad transformer is simply a law putting any (strong) monad into correspondence with some other (strong) monad over the same category.

Example 1.5. [Mog89a] Let $\mathbb{T} = (T, \eta^{\mathbb{T}}, \_^{\dagger})$ be a monad over a category $\mathbf{C}$. Let us denote by $\mathbb{P} = (P, \eta^{\mathbb{P}}, \_^{\ddagger}, \tau^{\mathbb{P}})$ a monad over $\mathbf{C}$ obtained from $\mathbb{T}$ by applying one of the following monad transformers.

1. *Exception monad transformer:* $PA = T(A+E), \eta^{\mathbb{P}}_A = \eta^{\mathbb{T}}_{A+E} \circ \mathrm{inl}$,
   $f^{\ddagger} = [f, \eta^{\mathbb{T}}_{B+E} \circ \mathrm{inr}]^{\dagger}$ for any $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{P}}}(A,B)$.
2. *State monad transformer:* $PA = S \to T(A \times S), \eta^{\mathbb{P}}_A = \lambda a.\, \lambda s.\, \eta^{\mathbb{T}}_{A \times S}\langle a, s\rangle$,
   $f^{\ddagger} = \lambda c.\, \lambda s.\, (\lambda \langle x, y\rangle.\, f(x)(y))^{\dagger}(c(s))$ for any $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{P}}}(A,B)$.
3. *Continuation monad transformer:* $PA = (A \to TR) \to TR$, $\eta^{\mathbb{P}}_A = \lambda a.\, \lambda c.\, c(a)$,
   $f^{\ddagger} = \lambda c.\, \lambda r.\, c(\lambda a.\, f(a)(r))$ for any $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{P}}}(A,B)$.
4. *Output monad transformer:* $PA = \mu X.\, T(A + O \times X)$, $\eta^{\mathbb{P}}_A = \eta^{\mathbb{T}}_{A+O\times PA} \circ \mathrm{inl}$,
   $f^{\ddagger} = \mu g.\, [f, \lambda\langle u, x\rangle.\, \eta^{\mathbb{T}}_{O\times PB}\, \mathrm{inr}\langle u, g(x)\rangle]^{\dagger}$ for any $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{P}}}(A,B)$.
5. *Input monad transformer:* $PA = \mu X.\, T(A + (I \to X))$, $\eta^{\mathbb{P}}_A = \eta^{\mathbb{T}}_{A+(I\to PA)} \circ \mathrm{inl}$,
   $f^{\ddagger} = \mu g.\, [f, \lambda h.\, \lambda u.\, \eta^{\mathbb{T}}_{I\to PB}\, \mathrm{inr}(g(h(u)))]^{\dagger}$ for any $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{P}}}(A,B)$.

Note that there is no obvious definition of a powerset monad transformer.

Monad transformers are not generally required to exhibit good categorical behavior, like being endofunctors, but sure enough usually they do. Some natural conditions that can be imposed on monad transformers can be found in [Mog89a], but in fact none of them can be harmlessly made part of the definition because any such condition can be falsified by some relevant monad example.

More specific than the notion of a monad transformer is the notion of monad morphism. We fix the category **Mnd** whose objects are monads over small categories and morphism are functors between the underlying categories coherently preserving monad structure in the following manner.

Definition 1.35. Given two monads $\mathbb{T} = (T, \eta^R, \mu^R)$ and $\mathbb{S} = (S, \eta^S, \mu^S)$ a *monad morphism* $\mathbb{T} \to \mathbb{S}$ consists of a functor $F : \mathbf{C} \to \mathbf{D}$ between the underlying categories and a natural transformation $\alpha : FT \to SF$ making the diagrams

$$\begin{tikzcd}
 & FT \arrow[dd, "\alpha"] \\
F \arrow[ur, "F\eta"] \arrow[dr, "\eta F"'] & \\
 & SF
\end{tikzcd}
\qquad
\begin{tikzcd}
 & SFT \arrow[r, "S\alpha"] & SSF \arrow[dd, "\mu F"] \\
FTT \arrow[ur, "\alpha T"] \arrow[dr, "F\mu"'] & & \\
 & FT \arrow[r, "\alpha"] & SF
\end{tikzcd}$$

commute. We call a monad morphism (F, α) a *monad embedding* if F is faithful and α_A is monic for every A.

Such a treatment of the category of monads is less general than the more abstract one of [Str72] presenting a monad as 2-categorical concept. The definition of monad morphism above is the same as in [Mog89a] and contrasts with the central definition of monad morphism from [Str72], where the morphisms from Definition 1.35 are called *opmorphisms*. Our choice of the definition of a monad morphism is justified by the following theorem, the first part of which is due to Moggi [Mog89a].

Theorem 1.36. *There is a one-to-one correspondence between monad morphisms (F, α) with (T, η^T, μ^T) over $\mathbf{C}$ as the source and (S, η^S, μ^S) over $\mathbf{D}$ as the target and pairs of functors $(F : \mathbf{C} \to \mathbf{D}, G : \mathbf{C}_{\mathbb{T}} \to \mathbf{D}_{\mathbb{T}})$ such that the following diagram commutes.*

$$\begin{tikzcd}
\mathbf{C} \arrow[r, "F"] \arrow[d, "K_{\mathbb{T}}"'] & \mathbf{D} \arrow[d, "K_{\mathbb{S}}"] \\
\mathbf{C}_{\mathbb{T}} \arrow[r, "G"] & \mathbf{D}_{\mathbb{S}}
\end{tikzcd}$$

where $K_{\mathbb{T}}$ and $K_{\mathbb{S}}$ are left adjoints by Kleisli construction. Moreover, under this correspondence monad embeddings are mapped to pairs of faithful functors.

Proof. Let (F, α) be a monad morphism. We define the functor $G : \mathbf{C}_{\mathbb{T}} \to \mathbf{D}_{\mathbb{T}}$ by putting $GA := FA$ for every $A \in \mathsf{Ob}(\mathbf{C}_{\mathbb{T}}) = \mathsf{Ob}(\mathbf{C})$ and $Gf := \alpha_B \circ Ff$ for every $f \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B) = \mathsf{Hom}_{\mathbf{C}}(A, TB)$. Since both $K_{\mathbb{T}}$ and $K_{\mathbb{S}}$ are identical on objects the functor equality stated by the diagram in question trivially holds on objects. On the other hand, for every $f \in \mathsf{Hom}_{\mathbf{C}}(A, B)$, $(GK_{\mathbb{T}})f = \alpha_B \circ F(\eta_B \circ f) = \alpha_B \circ F(\eta_B) \circ Ff = \eta_{FB} \circ Ff$ which proves the diagram for the case of morphisms.

Now suppose we are given a pair of functors $(F : \mathbf{C} \to \mathbf{D}, G : \mathbf{C}_{\mathbb{T}} \to \mathbf{D}_{\mathbb{T}})$, making the diagram commute. Note that this implies in particular that $GA = FA$ for every $A \in \mathsf{Ob}(\mathbf{C}) = \mathsf{Ob}(\mathbf{C}_{\mathbb{T}})$. Show how to construct a monad morphism (F, α). The missing part is the natural transformation $\alpha : FT \to SF$. For every $A \in \mathsf{Ob}(\mathbf{C})$ define $\alpha_A := G(\mathrm{id}_{TA})$. It is straightforward to verify that (F, α) makes a monad morphism.

Finally, observe that every monad embedding (F, α) gives rise to a pair of faithful functors. Functor F is faithful by definition. Show that G is also faithful. Let $f, g \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B) = \mathsf{Hom}_{\mathbf{C}}(A, TB)$. Then $Gf = Gg$ by definition means $\alpha_B \circ Ff = \alpha_B \circ Fg$. Since α_B is a monic, this is equivalent to $Ff = Fg$, i.e. to $f = g$ because F is faithful, and thus we are done. □

Theorem 1.36 assures that the chosen definition of a monad morphism is indeed the right one for our purposes because it gives rise to a functor between the corresponding Kleisli categories in contrast to the the dual notion from [Str72] which gives rise to a functor between the corresponding categories of Eilenberg-Moore algebras [PW02].

According to Moggi's program of using monads for denotational semantics a programming language with (unary) type constructors and polymorphic operators should be viewed as a free monad **FreeMnd**. Models of this programming language are supposed to be 2-functors from **FreeMnd** to **Cat**. It turns out that there is one-to-one correspondence between such models and monads in **Cat**. Moreover, this correspondence can be extended to a correspondence between model transformations (which are lax natural transformations) and monad morphisms. Following these lines further, it is natural to consider monad embeddings as transformations of language interpretations from models supporting less structure to ones supporting more. A perfect example of such an embedding is also given by Moggi. Namely, he has discovered (in accordance with earlier general observations by Lambek and Scott [LS86]) that any monad in **Mnd** can be embedded into a monad over the topos of presheaves. Moreover, if the source monad was strong, the target monad turns out to be strong as well. Essentially, this means the notion of the monad is relatively independent of the underlying category; in particular, one can always assume w.l.o.g. a rather rich categorical structure (including coproducts, exponentials, power objects, etc.) to be present.

Theorem 1.37. [Mog89b] *Let $\mathbf{C}$ be a small category, let $\widehat{\mathbf{C}} = \mathbf{Set}^{\mathbf{C}^{op}}$ be the corresponding topos of presheaves and let Y be the Yoneda embedding of $\mathbf{C}$ into $\widehat{\mathbf{C}}$. Then for every monad $\mathbb{T} = (T, \eta, \mu)$ there exists a monad $\widehat{\mathbb{T}} = (\widehat{T}, \hat{\eta}, \hat{\mu})$ over $\widehat{\mathbf{C}}$ such that the following diagram commutes*

$$\begin{tikzcd}
\mathbf{C} \arrow[r, "T"] \arrow[d, "Y"'] & \mathbf{C} \arrow[d, "Y"] \\
\widehat{\mathbf{C}} \arrow[r, "\widehat{T}"] & \widehat{\mathbf{C}}
\end{tikzcd}$$

i.e. (Y, id) is a monad embedding of $\mathbb{T}$ into $\widehat{\mathbb{T}}$, and for all $A \in \mathbf{C}$ the following equations hold

$$\hat{\eta}_{Y_A} = Y(\eta_A), \qquad \hat{\mu}_{Y_A} = Y(\mu_A).$$

Moreover, if $\mathbb{T}$ is strong then there exists a natural transformation $\hat{t}$, turning $\mathbb{T}$ into a strong monad, and for any $A, B \in \mathbf{C}$ satisfying the equation

$$\hat{\tau}_{YA,YB} = Y(\tau_{A,B})$$

As was remarked by Moggi, the result of such a lifting is not always what one expects, e.g. the expression defining the functorial part of the monad is not in general maintained. For example, the monad of partial computations given by $TX = X + 1$ gives rise to a monad with the same functorial part over the topos of presheaves but the state monad does not.

1.6 Contribution and related work

The content of the preceding section is of a preliminary nature. Its main goal was to introduce the language and the methods of dealing with computational monads. Many constructions presented here will be used as prototypes for more advanced cases in the following sections where we operate with further extensions of ME.

We have made precise the usage of rewriting technique for the metalanguage of effects, which justifies neatly the previous attempts [GSM06, MSG10]. The main technical achievement here is a proof of a strong normalisability theorem for a novel rewriting system completely capturing the identities of strong monads. This resulted in decidability of the equational theory of strong monads. The strong normalisation proof combines the ideas of [BBdP98] and [Cd96]. The related works featuring similar results include the following. The results of Benton, Bierman at al. [BBdP98] imply strong normalisation of a reduction system capturing only β-reductions of ME. Lindley and Stark [LS05] have shown strong normalisation of the metalanguage of effects but did not cover the unit type. An alternative canonical reduction system capturing the laws of strong monads but not the unit type is found in the dissertation of Gehrke [GTA95].

The restriction imposed on the operations of a plain signature is somewhat similar to the algebraicity requirement specific for the treatment of computational effects based on Lawvere theories (e.g. [PP02]). In both cases the role of the restriction is to rule out the possibility of introducing user-defined control operators, but the difference is in deciding what 'user-defined' means. In our settings, user-defined operators are those that possess counterparts in the underlying signature, whereas in terms of Lawvere theories, user-defined operators are those that are not induced by the monad functor.

Chapter 2

Additive monads

2.1 Introduction

Nondeterminism is a special kind of side-effect that is clearly of major importance. There are several reasons why nondeterminism deserves individual consideration. First of all, it can be combined with other side-effects abstractly, without appealing to concrete implementation: $p + q$ is just for any p and q, no matter what p and q do and what side-effects they have. (Unfortunately, this simple intuition contrasts with the fact that there is no evident machinery to impart nondeterminism to a given monad, such as a powerset monad transformer.) Second, nondeterminism, powered by nontermination, provides an expressive framework for encoding control structures both for nondeterministic and for deterministic computations, in particular via the Fischer-Ladner encoding of the `if` and `while` operators of imperative programming [FL79].

To be sure, the notion of nondeterminism is much too subtle to be ultimately captured by any concrete formalisation. Even when restricted to algebraic settings one may argue different variations of the set of equations specifying nondeterminism. It is therefore not surprising that there are different examples of monads implementing the idea of nondeterminism. E.g. a list monad can be considered as a monad with nondeterminism. In this case, the nondeterministic choice is not commutative, which still might be perfectly sensible (cf. e.g. [Hin00, KSFS05]).

The notion of nondeterminism used in this chapter (and the following ones) is rather strong – it is basically obtained by equationally axiomatising a powerset monad. The motivation behind this is as follows: first, this model of nondeterminism is better suited for theoretical concerns. It is abstract in the sense that it does not allow any cheating, e.g. one cannot grab the first program of a sum and execute it first, etc. Second, this notion is close to the notion accepted in the process algebra and hence might be useful for modelling side-effectful parallel processes. Third, it allows adequate encoding of the the `if` and `while` control primitives.

2.2 Soundness, completeness and decidability

The requirement for a monad to support nondeterminism amounts to some intuitively predictable conditions, imposed on its Kleisli category.

Definition 2.1 (Additive monad)**.** An *additive monad* is a monad $\mathbb{T}$, whose Kleisli category $\mathbf{C}_{\mathbb{T}}$ is enriched over join semilattices[1].

The presented definition comprises a great deal of information that is not immediately apparent. We unravel it as follows. By definition of the enriched category, every hom-set in $\mathbf{C}_{\mathbb{T}}$ must be equipped with the operations of join semilattices, that is, for every $A, B \in \mathrm{Ob}(\mathbf{C}_{\mathbb{T}})$ there is *bottom* $\odot_{A,B} \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B)$ and *join* $\oplus_{A,B} : \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B) \times \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B) \to \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B)$ such that $\mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B)$ makes a join-semilattice under them. Moreover, Kleisli composition $\diamond : \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(B,C) \times \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,B) \to \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A,C)$ must be bilinear with respect to $\oplus$ and respect $\odot$ on both sides, which results in the identities:

$$\begin{aligned}
\odot \diamond g &= \odot, & (f_1 \oplus f_2) \diamond g &= f_1 \diamond g \oplus f_2 \diamond g,\\
f \diamond \odot &= \odot, & f \diamond (g_1 \oplus g_2) &= f \diamond g_1 \oplus f \diamond g_2.
\end{aligned}$$

As usual, here and throughout we omit the indices at $\oplus$ and $\odot$. Also, we agree that $\diamond$ binds stronger than $\oplus$.

Unfortunately, the naive attempt to define a *strong additive monad* as a strong monad that happens to be additive is not satisfactory, because, in general, strength does not respect the hom-set structure. The correct definition is as follows.

Definition 2.2 (Strong additive monad)**.** A *strong additive monad* is an additive monad that is strong and whose tensorial strength τ satisfies two extra conditions:

$$\tau_{A,B}\langle \mathrm{id}_A, \odot\rangle = \odot, \tag{2.1}$$

$$\tau_{A,B}\langle \mathrm{id}_A, f \oplus g\rangle = \tau_{A,B}\langle \mathrm{id}_A, f\rangle \oplus \tau_{A,B}\langle \mathrm{id}_A, g\rangle \tag{2.2}$$

for any morphisms $f, g \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A,B)$.

Remark 2.3. In the case of strong additive monads, the operations $\oplus$ and $\odot$ give rise to natural transformations $\varpi_A : TA \times TA \to TA$ and $\delta_A : C_1 A \to TA$ where C_1 is a constant functor, mapping everything to the terminal object 1. Indeed, let us define $\varpi_A := \pi_1 \oplus \pi_2$ and $\delta_A := \odot$ where π_i $(i = 1, 2)$ are left and right cartesian projections from $TA \times TA$ to TA. Let $f \in \mathsf{Hom}_{\mathbf{C}}(A,B)$. Then the naturality of ϖ and δ can be

[1]We assume that a join semilattice is always *bounded*, i.e. it comes equipped with the bottom element.

expressed by the commutativity diagrams:

$$\begin{tikzcd} TA \times TA \arrow[r, "\varpi_A"] \arrow[d, "Tf\times Tf"'] & TA \arrow[d, "Tf"] \\ TB \times TB \arrow[r, "\varpi_B"] & TB \end{tikzcd} \qquad \begin{tikzcd} C_1A = 1 \arrow[r, "\delta_A"] \arrow[d, "!"'] & TA \arrow[d, "Tf"] \\ C_1B = 1 \arrow[r, "\delta_B"] & TB \end{tikzcd}$$

The proof of the left diagram is as follows:

$$\begin{aligned} Tf \circ \varpi_A &= Tf \circ (\pi_1 \oplus \pi_2) = \mu \circ T(\eta \circ f) \circ (\pi_1 \oplus \pi_2) \\ &= (\eta \circ f) \diamond (\pi_1 \oplus \pi_2) = (\eta \circ f) \diamond \pi_1 \oplus (\eta \circ f) \diamond \pi_2 \\ &= Tf \circ \pi_1 \oplus Tf \circ \pi_2 = \varpi_B \circ (Tf \times Tf). \end{aligned}$$

Analogously, the calculation: $Tf \circ \delta_A = Tf \circ \odot \circ ! = (\eta \circ f) \diamond \odot \circ ! = \odot \circ ! = \delta_B \circ !$ proves the commutativity of the second diagram.

It is useful to have equivalents of the conditions (2.1), (2.2) in terms of Kleisli categories. Let us define a family of operators $\Delta_{A,B} : \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B) \to \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, A \times B)$ by putting $\Delta_{A,B}(f) := \tau_{A,B}\langle \mathrm{id}, f\rangle$ for every $f \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B) = \mathsf{Hom}_{\mathbf{C}}(A, TB)$. Observe that, conversely, we can extract strength from Δ due to the equation

$$\tau_{A,B} = T(\mathrm{fst} \times \mathrm{id})(\Delta_{A\times TB,B}(\mathrm{snd})) \tag{2.3}$$

easily provable by Lemma 1.13.

Lemma 2.4. *A strong monad is strongly additive iff it is additive and all the $\Delta_{A,B}$ are homomorphisms of bounded join-semilattices, i.e. $\Delta(\odot) = \odot$ and $\Delta(f \oplus g) = \Delta(f) \oplus \Delta(g)$ for all appropriate f, g.*

Proof. By definition, $\Delta(\odot) = \tau\langle \mathrm{id}, \odot\rangle$, therefore (2.1) is equivalent to $\Delta(\odot) = \odot$. On the other hand, for any appropriate f, g, $\Delta(f \oplus g) = \tau\langle \mathrm{id}, f \oplus g\rangle$, $\Delta(f) = \tau\langle \mathrm{id}, f\rangle$, $\Delta(g) = \tau\langle \mathrm{id}, g\rangle$ and thus (2.2) is equivalent to $\Delta(f \oplus g) = \Delta(f) \oplus \Delta(g)$. □

Prominent examples of strong additive monads include the powerset monad $\mathcal{P}$ and its variants possessing specific cardinality restrictions: the finitary powerset monad $\mathcal{P}_{fin}$, the countable powerset monad $\mathcal{P}_{\omega}$, etc. Other examples of additive monads can be obtained by adding other computational effects. Standardly, one can try to use monad transformers to generate new additive monads from existing ones, and indeed many monad transformers turn out to preserve additivity.

Theorem 2.5. *Let $\mathbb{T} = (T, \eta, \_^{\dagger}, \tau)$ be a strong additive monad over a cartesian category $\mathbf{C}$. We denote bottom and join of $\mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B)$ by $\odot^{\mathbb{T}}_{A,B}$ and $\oplus^{\mathbb{T}}_{A,B}$. Then a monad $\mathbb{R}$ obtained by applying any of the following monad transformers to $\mathbb{T}$ is strongly additive with respect to to the join semilattice operators $\odot^{\mathbb{R}}_{A,B}$ and $\oplus^{\mathbb{R}}_{A,B}$ defined hereby.*

1. *State monad transformer* $RA = S \to T(A \times S)$ *with* $\odot^{\mathbb{R}}_{A,B} := \lambda s.\, \odot^{\mathbb{T}}_{A,B}$ *and* $f \oplus^{\mathbb{R}}_{A,B} g := \lambda a.\, \big(f(a) \oplus^{\mathbb{T}}_{S,B\times S} g(a)\big)$ *for every* $f, g \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{R}}}(A, B)$.
2. *Continuation monad transformer* $RA = (A \to TK) \to TK$ *with* $\odot^{\mathbb{R}}_{A,B} := \lambda c.\, \odot^{\mathbb{T}}_{A,K}$ *and* $f \oplus^{\mathbb{R}}_{A,B} g := \lambda c.\, f(c) \oplus^{\mathbb{T}}_{A,K} g(c)$ *for every* $f, g \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{R}}}(A, B)$.

Proof. It is well known that both monad transformers send strong monads to strong monads (cf. e.g. [Mog89a]). We are left to verify strong additivity.

1. Observe that for every $A, B \in \mathsf{Ob}\,(\mathbf{C})$, $\mathsf{Hom}_{\mathbf{C}_{\mathbb{R}}}(A, B) = \mathsf{Hom}_{\mathbf{C}}(A, S \to T(B \times S)) \cong \mathsf{Hom}_{\mathbf{C}}(A \times S, T(B \times S)) = \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A \times S, B \times S)$ which means that the functor J from $\mathbf{C}_{\mathbb{R}}$ to $\mathbf{C}_{\mathbb{T}}$ sending every object $A \in \mathsf{Ob}\,(\mathbf{C}_{\mathbb{R}})$ to $(A \times S) \in \mathsf{Ob}\,(\mathbf{C}_{\mathbb{T}})$ and every morphism $f \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{R}}}(A, B)$ to $(\lambda\langle a, s\rangle.\, f(a)(s)) \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A \times S, B \times S)$ is a full and faithful embedding. Therefore, the category $\mathbf{C}_{\mathbb{R}}$ is also enriched over join semilattices, i.e. $\mathbb{R}$ is an additive monad with respect to the join semilattice structure induced by this embedding. It is easy to see that this structure is the same as we have explicitly defined.

   For every $A, B \in \mathbf{C}_{\mathbb{R}}$ we define $\Delta^{\mathbb{R}}_{A,B}$ by putting $\Delta^{\mathbb{R}}_{A,B}(f) := J^{-1}(\Delta^{\mathbb{T}}_{S\times A, S\times B}(J(f)))$, for all $f \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{R}}}(A, B)$. By (2.3) for every A, B we can define $\tau^{\mathbb{R}}_{A,B}$. It is easy to see that $\tau^{\mathbb{R}}$ is indeed strength, making $\mathbb{R}$ a strong additive monad.
2. Analogous to the previous case. □

Remark 2.6. If we instantiate O in the output monad transformer by 1 or, equivalently, if we instantiate I in the input monad transformer by 1 we obtain the same result: $RA = \mu X.T(A + X)$, called a *resumption monad transformer* (cf. e.g. [CM93]). This monad transformer is of independent interest mainly because it can be used for modelling concurrent processes. In particular, one can recursively define a parallel composition of two programs $p, q : RA$ as the least solution of the system of equations:

$$\begin{aligned} p \parallel q &= p \mathbin{\lfloor\!\lfloor} q + q \mathbin{\lfloor\!\lfloor} p, \\ p \mathbin{\lfloor\!\lfloor} q &= \mathrm{do}\ x \leftarrow p; \mathrm{ret}(p \parallel q). \end{aligned}$$

Here $+$ is an operation symbol whose intended interpretation is $\oplus$. Using the fact that RA makes an additive monad, as proved in Theorem 2.5, one can establish a great deal of analogues of process algebra laws, such as $(p + q) \mathbin{\lfloor\!\lfloor} r = (p \mathbin{\lfloor\!\lfloor} r) + (q \mathbin{\lfloor\!\lfloor} r)$, etc. A remarkable example of an identity that cannot be proved for monad R is the right distributivity law:

$$\mathrm{do}\ x \leftarrow p; (q + r) = \mathrm{do}\ x \leftarrow p; q + \mathrm{do}\ x \leftarrow p; r.$$

This completely agrees with the process algebra parallels, but since this law is part of strong additive monad axiomatisation (Fig 2.1), we conclude that the resumption monad transformer does **not** preserve additivity. Hence both input and output monad transformers also do not.

A remarkable example of a monad transformer **not** preserving additivity is that the exception monad transformer $\mathcal{P}(\_ + E)$ is **not** an additive monad — the only violated law is **(dist$_1^+$)**, e.g. if a program r throws an exception then $(\mathrm{do}\ x \leftarrow r; \varnothing) = r \neq \varnothing$. An intuitive reason why some monad transformers do not preserve additivity is because additive monads provide a control mechanism consisting of nondeterminism and nontermination, and if the monad transformer also provides some sort of control they often get into conflict.

The internal language of strong additive monads extends the metalanguage of effects defined in Chapter 1 by the operations *choice* and *deadlock*:

$$\frac{}{\Gamma \rhd \varnothing : TA} \qquad\qquad \frac{\Gamma \rhd p : TA \quad \Gamma \rhd q : TA}{\Gamma \rhd p + q : TA}$$

whose semantics is provided by the assignments:

$$\begin{aligned} [\![\Gamma \rhd \varnothing : TA]\!] &:= \odot_{[\![\Gamma]\!],[\![A]\!]}, \\ [\![\Gamma \rhd p + q : TA]\!] &:= [\![\Gamma \rhd p : TA]\!] \oplus_{[\![\Gamma]\!],[\![A]\!]} [\![\Gamma \rhd q : TA]\!]. \end{aligned}$$

Essentially, we deal with the same language as in Chapter 1: $\varnothing$ and $+$ can be considered as families of distinguished signature symbols from Σ indexed by the argument types (omitted at writing). The calculus of additive monads ME^+ is obtained from ME by adding the axioms given in Fig 2.1 and the evident congruence rules **(cong_nil)** and **(cong_plus)**. The laws **(assoc$^+$)**, **(comm$^+$)** and **(idmp$^+$)** will be referred to as ACI-*laws*. Let $\equiv$ be the least congruence encompassing them. Programs, related by $\equiv$ will be occasionally called ACI-*equal*. The choice operator immediately gives rise to the partial order $\leqslant$, defined by the equivalence $p \leqslant q \iff p + q = q$.

We call a program *deterministic* if it contains neither $\varnothing$ nor $+$. In the sequel we will occasionally use the notation $\sum_{i \in I} p_i$ where I is a finite set, e.g. $I = \{i_1, \ldots, i_n\}$ as a shortening for $(p_{i_1} + \ldots + p_{i_n})$ if $n > 0$ and $\varnothing$ if $n = 0$. We write $\sum_i p_i$ instead of $\sum_{i \in I} p_i$ if I is irrelevant or implied by the context. Note that $\sum_i p_i$ is defined up to associativity.

Theorem 2.7 (Soundness and completeness)**.** *The calculus* ME^+ *is sound and strongly complete over strong additive monads.*

Proof. Soundness. It suffices to prove soundness of the newly introduced rules. The claim is straightforward for **(nil$^+$)**, **(comm$^+$)**, **(idmp$^+$)** and **(assoc$^+$)**. Let us show the remainder.

(nil$^+$)	$p + \varnothing = p$	**(comm$^+$)**	$p + q = q + p$
(idmp$^+$)	$p + p = p$	**(assoc$^+$)**	$p + (q + r) = (p + q) + r$
(dist$_1^\varnothing$)	$\mathrm{do}\ x \leftarrow \varnothing; r = \varnothing$	**(dist$_2^\varnothing$)**	$\mathrm{do}\ x \leftarrow r; \varnothing = \varnothing$

(dist$_1^+$) $\quad \mathrm{do}\ x \leftarrow (p + q); r = \mathrm{do}\ x \leftarrow p; r + \mathrm{do}\ x \leftarrow q; r$

(dist$_2^+$) $\quad \mathrm{do}\ x \leftarrow r; (p + q) = \mathrm{do}\ x \leftarrow r; p + \mathrm{do}\ x \leftarrow r; q$

FIGURE 2.1: Axioms of additive monads

Axiom **(dist$_1^\varnothing$)**:

$$\begin{aligned}
&[\![\Gamma \rhd\ \mathrm{do}\ x \leftarrow \varnothing; r : TB]\!] \\
&\quad = [\![\Gamma, x : A \rhd r : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}, [\![\Gamma \rhd \varnothing : TA]\!] \rangle && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma, x : A \rhd r : TB]\!] \diamond \odot = \odot = [\![\Gamma \rhd \varnothing : TB]\!]. && \text{[by 2.1, def. of } [\![\_]\!], \odot]
\end{aligned}$$

Axiom **(dist$_2^\varnothing$)**:

$$\begin{aligned}
&[\![\Gamma \rhd\ \mathrm{do}\ x \leftarrow r; \varnothing : TB]\!] \\
&\quad = [\![\Gamma, x : A \rhd \varnothing : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ [\![\Gamma \rhd r : TA]\!] && \text{[by def. of } [\![\_]\!]] \\
&\quad = \odot \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ [\![\Gamma \rhd r : TA]\!] = \odot = [\![\Gamma \rhd \varnothing : TB]\!]. && \text{[by def. of } [\![\_]\!], \odot]
\end{aligned}$$

Axiom **(dist$_1^+$)**:

$$\begin{aligned}
&[\![\Gamma \rhd\ \mathrm{do}\ x \leftarrow (p + q); r : TB]\!] \\
&\quad = [\![\Gamma, x : A \rhd r : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \\
&\qquad \langle \mathrm{id}, [\![\Gamma \rhd p : TA]\!] \oplus [\![\Gamma \rhd q : TA]\!] \rangle && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma, x : A \rhd r : TB]\!] \diamond (\tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}, [\![\Gamma \rhd p : TA]\!] \rangle \oplus \\
&\qquad \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}, [\![\Gamma \rhd q : TA]\!] \rangle) && \text{[by 2.2]} \\
&\quad = [\![\Gamma, x : A \rhd r : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}, [\![\Gamma \rhd p : TA]\!] \rangle \oplus \\
&\qquad [\![\Gamma, x : A \rhd r : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ \langle \mathrm{id}, [\![\Gamma \rhd q : TA]\!] \rangle && \text{[by def. of } \oplus] \\
&\quad = [\![\Gamma \rhd \mathrm{do}\ x \leftarrow p; r : TB]\!] \oplus [\![\Gamma \rhd \mathrm{do}\ x \leftarrow q; r : TB]\!] && \text{[by def. of } [\![\_]\!]] \\
&\quad = [\![\Gamma \rhd \mathrm{do}\ x \leftarrow p; r + \mathrm{do}\ x \leftarrow q; r : TB]\!]. && \text{[by def. of } [\![\_]\!]]
\end{aligned}$$

Axiom **(dist$_2^+$)**:

$$\begin{aligned}
&[\![\Gamma \rhd\ \mathrm{do}\ x \leftarrow r; (p + q) : TB]\!] \\
&\quad = ([\![\Gamma, x : A \rhd p : TB]\!] \oplus [\![\Gamma, x : A \rhd p : TB]\!]) \diamond \\
&\qquad \tau_{[\![\Gamma]\!],[\![A]\!]} \circ [\![\Gamma \rhd r : TA]\!] && \text{[by def. of } [\![\_]\!]]
\end{aligned}$$

$$\begin{aligned}
&= [\![\Gamma, x : A \rhd p : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ [\![\Gamma \rhd r : TA]\!] \oplus \\
&\quad [\![\Gamma, x : A \rhd p : TB]\!] \diamond \tau_{[\![\Gamma]\!],[\![A]\!]} \circ [\![\Gamma \rhd r : TA]\!] && \text{[by def. of } \oplus] \\
&= [\![\Gamma \rhd \mathrm{do}\ x \leftarrow r; p : TB]\!] \oplus [\![\Gamma \rhd \mathrm{do}\ x \leftarrow r; q : TB]\!] && \text{[by def. of } [\![\_]\!]] \\
&= [\![\Gamma \rhd \mathrm{do}\ x \leftarrow r; p + \mathrm{do}\ x \leftarrow r; q : TB]\!]. && \text{[by def. of } [\![\_]\!]]
\end{aligned}$$

Completeness. We extend the syntactic construction from Theorem 1.16 where we have build a strong monad $\mathbb{T}_{\Sigma,\Phi}$. A strong additive monad $\mathbb{P}_{\Sigma,\Phi}$ is obtained by strengthening the underlying equivalence relation $\sim$ up to provable equivalence in ME^+. We define

$$\begin{gathered}
\odot_{A,B} := [x : A \rhd \varnothing : TB]_\sim, \\
[x : A \rhd p : TB]_\sim \ \oplus_{A,B}\ [x : A \rhd p : TB]_\sim := [x : A \rhd p + q : TB]_\sim.
\end{gathered}$$

By virtue of the proof of Theorem 1.16 we only need to show that so defined strong monad $\mathbb{P}_{\Sigma,\Phi}$ is indeed a strong additive monad.

It is easy to see that $\oplus$ and $\odot$ satisfy all the axioms of join semilattices – basically, all of them appear explicitly in the calculus. E.g., commutativity follows from **(comm$^+$)**: $[x : A \rhd p : TB]_\sim \oplus [x : A \rhd q : TB]_\sim = [x : A \rhd p + q : TB]_\sim = [x : A \rhd q + p : TB]_\sim = [x : A \rhd q : TB]_\sim \oplus [x : A \rhd p : TB]_\sim$. Similarly, using the axioms **(dist$_2^+$)**, **(dist$_1^+$)**, **(dist$_1^\varnothing$)**, **(dist$_2^\varnothing$)**, one can prove distributivity of binding over finite joins.

Finally, the proof of (2.1) and (2.2) run as follows:

$$\begin{aligned}
&\tau_{A,B}\langle \mathrm{id}_A, \odot \rangle \\
&\quad = [(x : A, p : B) \rhd \mathrm{do}\ y \leftarrow \varnothing; \mathrm{ret}\langle x, y\rangle]_\sim = [\varnothing]_\sim = \odot, && \text{[by } \mathbf{(dist_1^\varnothing)}]
\end{aligned}$$

$$\begin{aligned}
&\tau_{A,B}\langle \mathrm{id}_A, [x : A \rhd p : TB]_\sim \oplus [x : A \rhd q : TB]_\sim \rangle \\
&\quad = [(x : A, p : B) \rhd \mathrm{do}\ y \leftarrow (p + q); \mathrm{ret}\langle x, y\rangle]_\sim \\
&\quad = [(x : A, p : B) \rhd \mathrm{do}\ y \leftarrow p; \mathrm{ret}\langle x, y\rangle]_\sim \oplus \\
&\qquad [(x : A, p : B) \rhd \mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x, y\rangle]_\sim && \text{[by } \mathbf{(dist_1^+)}] \\
&\quad = \tau_{A,B}\langle \mathrm{id}_A, [x : A \rhd p : TB]_\sim \rangle \oplus \tau_{A,B}\langle \mathrm{id}_A, [x : A \rhd q : TB]_\sim \rangle.
\end{aligned}$$

The proof of the theorem is thus completed. □

Theorem 2.7 justifies the use of the computational metalanguage for strong additive monads. From now onwards we can use it safely for all kinds of arguments instead of the language of morphisms and commutative diagrams. Our next milestone is proving that the theory of strong additive monads is decidable. To that end, we complete the reduction system in page 26 by the following rules:

σ-rules:

$$\begin{array}{rclcrcl} p + \varnothing & \rightarrowtail & p & \quad & \text{do } x \leftarrow p; \varnothing & \rightarrowtail & \varnothing \\ \varnothing + p & \rightarrowtail & p & \quad & \text{do } x \leftarrow \varnothing; p & \rightarrowtail & \varnothing \end{array}$$

$$\begin{array}{rcl} \text{do } x \leftarrow (p+q); r & \rightarrowtail & \text{do } x \leftarrow p; r + \text{do } x \leftarrow q; r \\ \text{do } x \leftarrow p; (q+r) & \rightarrowtail & \text{do } x \leftarrow p; q + \text{do } x \leftarrow p; r \end{array}$$

We refer to the top two lines of this system as σ_0-rules and to the two bottom lines by σ_1-rules. Hence, $\rightarrowtail_\sigma = (\rightarrowtail_{\sigma_0} \cup \rightarrowtail_{\sigma_1})$.

Lemma 2.8. *The combined reduction relation $\rightarrowtail_{\beta\eta\sigma}$ is strongly normalising with respect to $\star$-normal programs iff the combined reduction relation $\rightarrowtail_{\beta\sigma_1}$ is too.*

Proof. In perfect analogy with the proof of Lemma 1.23 one can easily ensure that $\rightarrowtail_{\beta\sigma_1}$ quasi-commutes over $\rightarrowtail_{\eta\sigma_0}$. As the relation $\rightarrowtail_{\eta\sigma_0}$ is size-decreasing, and hence terminating, by Lemma 1.19 this implies the claim. □

The relation $\rightarrowtail_{\beta\sigma_1}$ and thus (by Lemma 2.8) $\rightarrowtail_{\beta\eta\sigma}$ indeed turn out to be strongly normalising with respect to $\star$-normal programs. A strict proof of this fact calls for some background knowledge concerning multisets. First, we adopt the following notation. Let us denote by $\wr a_1, \ldots, a_n \int$ the multiset, consisting of elements $a_1, \ldots, a_n$ (analogous to $\{a_1, \ldots, a_n\}$ in the case of sets). In particular, $\wr\ \int$ shall denote the empty multiset. As in the case of sets, the comprehension construction $\wr f(x_1, \ldots, x_n) \mid x_1 \in a_1, \ldots, x_n \in a_n \int$ forms a new multiset out of the multisets $a_1, \ldots, a_n$ by replacement, with x_i, ranging over elements of a_i in respect of multiplicity. For example, $\wr \text{do } x; y \mid x \in \wr a, a \int, y \in \wr a, b \int \int$ shall denote $\wr (\text{do } a; a), (\text{do } a; b), (\text{do } a; a), (\text{do } a; b) \int$. No other special notation for multisets shall be used. Instead, the usual operators over sets shall be overridden, e.g. $s \cup t$, with s and t being multisets, shall denote the multiset union, i.e. the union respecting multiplicity.

Given a strict poset $(A, \succ)$, one can derive a strict partial order $\succ_m$ over multisets with elements in A as the closure under multiset union and transitivity of the clauses

$$\wr a \int \succ_m \wr b_1, \ldots, b_n \int \quad \text{if } a \succ b_i \quad \text{for all } i.$$

A core result from this ordering is the following lemma, originally proved in [DM79].

Lemma 2.9. *If the partial order $\succ$ is well-founded then so is $\succ_m$.*

Let us introduce a recursive operation $\mathfrak{m}$ taking single programs to finite multisets of (+-free) programs by the equation $\mathfrak{m}(p+q) = \mathfrak{m}(p) \cup \mathfrak{m}(q)$ and distributing over the remaining term constructors, e.g. $\mathfrak{m}(\text{do } x \leftarrow p; q) = \wr \text{do } x \leftarrow s; t \mid s \in \mathfrak{m}(p), t \in \mathfrak{m}(q) \int$. Essentially, $\mathfrak{m}$ being applied to a program lets all the term constructors distribute over choice (even though this is semantically unsound) and then converts the obtained grand choice into a multiset. Operator $\mathfrak{m}$ extends to contexts in the obvious manner, resulting in a multiset of contexts. Given any context C, we denote by $\mathfrak{m}_0(C)$ the part of $\mathfrak{m}(C)$ consisting of the contexts not containing the hole, i.e. essentially of programs and by $\mathfrak{m}_1(C)$ the remaining contexts with precisely one hole. By definition, $\mathfrak{m}(C) = \mathfrak{m}_0(C) \cup \mathfrak{m}_1(C)$.

Lemma 2.10. *For any program t and any context C:*

$$\mathfrak{m}(C\{t\}) = \wr D\{s\} \mid D \in \mathfrak{m}_1(C), s \in \mathfrak{m}(t) \int \cup \mathfrak{m}_0(C).$$

Unfortunately, the naive guess that $\mathfrak{m}(C\{t\}) = \wr D\{s\} \mid D \in \mathfrak{m}(C), s \in \mathfrak{m}(t) \int$ in not generally correct. Specifically, it can be falsified by the example: $C := \Box + r$, $t := p + q$. Indeed, we have:

$$\mathfrak{m}(C\{t\}) = \mathfrak{m}((p+q)+r) = \wr p,q,r \int \neq \wr p,q,r,r \int = \wr D\{s\} \mid D \in \wr \Box, r \int, s \in \wr p,q \int \int.$$

Proof. Induction over the term complexity of C. The claim is trivial if $C \doteq \Box$ or if C does not contain $\Box$. Otherwise we proceed by case analysis.

– Let the topmost constructor of C be not choice and let $C \doteq M\{K\}$, where K is the maximal context distinct from C. Equivalently, K is the maximal proper subterm of C containing $\Box$, and hence it is uniquely defined. Then by definition:

$$\begin{aligned}
\mathfrak{m}(C\{t\}) &= \mathfrak{m}(M\{K\{t\}\}) \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(M), s \in \mathfrak{m}(K\{t\}) \int \\
&= \wr D\{E\{s\}\} \mid D \in \mathfrak{m}_1(M), E \in \mathfrak{m}_1(K), s \in \mathfrak{m}(t) \int \cup \\
&\quad \wr D\{s\} \mid D \in \mathfrak{m}_1(M), s \in \mathfrak{m}_0(K) \int && \text{[by ind.]} \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(M\{K\}), s \in \mathfrak{m}(t) \int \cup \mathfrak{m}_0(M\{K\}) \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(C), s \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(C).
\end{aligned}$$

– Let $C = K + r$. Then

$$\begin{aligned}
\mathfrak{m}(C\{t\}) &= \mathfrak{m}(K\{t\}) \cup \mathfrak{m}(r) \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(K), s \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(K) \cup \mathfrak{m}(r) && \text{[by ind.]} \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(K+r), s \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(K+r) \\
&= \wr D\{s\} \mid D \in \mathfrak{m}_1(C), s \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(C).
\end{aligned}$$

– Let $C = r + K$. It follows by symmetry from the previous case. □

Let us denote by $|t|_x$ the number of occurrences of the variable x in term t.

Definition 2.11 (Linear and simple programs)**.** A program t is *linear in x* if for any $r \in \mathfrak{m}(t)$, $|r|_x \leqslant 1$ and *simple* if for every subterm of it of the form $(\mathsf{do}\ x \leftarrow p; q)$, q is linear in x.

For example, $(\mathsf{do}\ x \leftarrow p; (x + x))$ is a simple program, but $(\mathsf{do}\ x \leftarrow p; x; x)$ is not.

Lemma 2.12. *Let p and q be two simple programs and let q be linear in some variable x. Then the program $q[p/x]$, if well-defined, is also simple.*

Proof. Suppose, p, q are simple and q is linear in x. W.l.o.g. x does not occur in p. If q does not depend on x then we are trivially done. Otherwise we proceed by induction over n, the number of free occurrences of x in q. Let C be the context, obtained from q by replacing the last occurrence of x by $\Box$. Then, of course, $q[p/x] \doteq C\{p\}[p/x]$. Since $C\{p\}$ contains at most $n-1$ free occurrences of x, we will be done by induction hypothesis once we prove that $C\{p\}$ is simple and linear in x. Observe that by Lemma 2.10:

$$\mathfrak{m}(C\{p\}) = \wr D\{w\} \mid D \in \mathfrak{m}_1(C), w \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(C).$$

Since $x \notin \mathrm{Vars}(p)$, for any $t \in \mathfrak{m}(C\{p\})$, $|t|_x \leqslant \max\{\max_{D \in \mathfrak{m}_1(C)} |D|_x, \max_{D \in \mathfrak{m}_0(C)} |D|_x\} = \max_{D \in \mathfrak{m}(C)} |D|_x \leqslant \max_{s \in \mathfrak{m}(q)} |s|_x \leqslant 1$ and thus $C\{p\}$ is linear in x. We are left to prove the simplicity of $C\{p\}$.

Let $(\mathsf{do}\ z \leftarrow u; r)$ be a subterm of $C\{p\}$ and show that r is linear in z. Note that if $(\mathsf{do}\ z \leftarrow u; r)$ is a proper subterm of p then we are done by assuming the simplicity of p. Otherwise C has one of the forms $K\{\mathsf{do}\ z \leftarrow M; r\}$ or $K\{\mathsf{do}\ z \leftarrow u; M\}$. In the former case, r is linear in z by the definition of C. In the latter case, observe that p may not depend on z because $C\{p\}[p/x] \doteq K\{\mathsf{do}\ z \leftarrow u; M\{p\}\}[p/x] \neq q[p/x]$. Since $q \doteq C\{x\} = K\{\mathsf{do}\ z \leftarrow u; M\{x\}\}$ is simple, $M\{x\}$ is linear in z. By Lemma 2.10:

$$\begin{aligned}\mathfrak{m}(M\{p\}) &= \wr D\{w\} \mid D \in \mathfrak{m}_1(M), w \in \mathfrak{m}(p) \int \cup \mathfrak{m}_0(M),\\ \mathfrak{m}(M\{x\}) &= \wr D\{w\} \mid D \in \mathfrak{m}_1(M), w \in \mathfrak{m}(x) \int \cup \mathfrak{m}_0(M)\end{aligned}$$

from which it is clear by definition that the linearity of $M\{x\}$ in z and the fact that $z \notin \mathrm{Vars}(p)$ imply the linearity of $r \doteq M\{p\}$ in z. Therefore r is linear in z. □

Lemma 2.13. *Let $s \rightarrowtail_{\beta\sigma_1} t$ for some programs s and t. Then if s is simple, t is also simple. If s is moreover linear in some variable z, then so is t.*

Proof. First we prove the lemma if $s \rightarrowtail t$ is a rule instance. As usual, the most involved is the case of the second unit law $(\mathsf{do}\ x \leftarrow \mathsf{ret}\, p; q) \rightarrowtail q[p/x]$. Let us consider it in detail and drop the remaining rules, whose verification runs analogically.

Suppose $(\mathsf{do}\ x \leftarrow \mathsf{ret}\, p; q)$ is simple; in particular, q is linear in x. Then by Lemma 2.12 $q[p/x]$ is also simple. Now suppose that $(\mathsf{do}\ x \leftarrow \mathsf{ret}\, p; q)$ is moreover linear in some variable z. In particular, p and q cannot both contain z, because otherwise there would exist $u \in \mathfrak{m}(p)$ and $r \in \mathfrak{m}(q)$ such that $|u|_z \geqslant 1$, $|r|_z \geqslant 1$ and thus for $(\mathrm{do}\ x \leftarrow \mathrm{ret}\, u; r) \in \mathfrak{m}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q)$ we would have $|\,\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q|_z \geqslant 2$, contradicting linearity in z. As a result, every $r \in \mathfrak{m}(q[p/x]) = \mathfrak{m}(C\{p\}) = \wr D\{u\} \mid D \in \mathfrak{m}_1(C), u \in \mathfrak{m}(p) \wr \cup \mathfrak{m}_0(C)$ contains not more than one occurrence of z, i.e. by definition, $q[p/x]$ is linear in z.

Finally, let us consider the general case, i.e. for some context C and a $\beta\sigma_1$-rule $p \rightarrowtail q$, $s \doteq C\{p\}$ and $t \doteq C\{q\}$. Suppose s is simple and linear in some variable z. We prove that t is also simple and linear in z by induction over the complexity of C. The base case $C \doteq \Box$ is the one we have proved above. If C does not contain $\Box$ then $t \doteq s$ and hence t is simple and linear in z. Let us proceed with the induction step under the assumption that C does contain $\Box$. Let $C \doteq M\{K\}$ be a decomposition of C where K is chosen to be the maximal proper subterm of C containing $\Box$. By induction, $K\{q\}$ is simple and linear in z. Note that C can be obtained from K by applying precisely one term-formation rule. It is clear that all these rules, except the rule for binding, preserve simplicity and linearity in z, i.e. we will be done once we prove the following two cases.

- $M \doteq (\mathrm{do}\ y \leftarrow \Box; r)$ with some program r. By assumption, $s \doteq (\mathrm{do}\ y \leftarrow K\{p\}; r)$ is simple and linear in z. Therefore, both $K\{p\}$ and r are simple, linear in z, and moreover r is linear in y. By induction hypothesis, $K\{q\}$ is simple and linear in z. Therefore, by definition, $t \doteq (\mathrm{do}\ y \leftarrow K\{q\}; r)$ is simple. Note that z cannot occur both in $K\{p\}$ and in r — the opposite would contradict the linearity of s in z. Since the set of free variables cannot grow at rewriting, z cannot occur both in $K\{q\}$ and in r. Together with the other facts, this results in the linearity of t in z.
- $M \doteq (\mathrm{do}\ y \leftarrow u; \Box)$ with some program u. By assumption $s \doteq (\mathrm{do}\ y \leftarrow u; K\{p\})$ is simple and linear in z. Therefore, both $K\{p\}$ and r are simple, linear in z, and moreover $K\{p\}$ is linear in y. By induction hypothesis, $K\{q\}$ is simple and linear both in z and in y. By definition, $t \doteq (\mathrm{do}\ y \leftarrow u; K\{q\})$ is simple. By the same argument as in the previous clause, only one of the u, $K\{q\}$ may contain z, and therefore t is linear in z.

If one only assumes that s is simple without requiring linearity, then the simplicity of t can be shown by introducing a false variable z. □

Lemma 2.14. *The reduction relation $\rightarrowtail_{\beta\sigma_1}$ is strongly normalising with respect to simple programs.*

Proof. Both relations $\rightarrowtail_\beta$ and $\rightarrowtail_{\sigma_1}$ taken individually are strongly normalising. The proof of this fact for the first relation is certified by Theorem 1.24. We prove the strong normalisation property of the second one by picking out an appropriate polynomial interpretation (cf. e.g. [BN98]). Let us put into correspondence with every monad operator a polynomial of the corresponding arity over positive naturals as follows.

$$P(\varnothing) := 1, \qquad P(+) := X_1 + X_2, \qquad P(\mathrm{do}\ x \leftarrow \_\,;\_\,) := X_1^2 \cdot X_2^2.$$

This extends in a standard way to all terms built from deadlock, choice, binding and variables (P can be defined over variables arbitrarily). In order to prove that the relation $\rightarrowtail_{\sigma_1}$ is terminating, it suffices to make sure that for every σ_1-rule the polynomial interpretation of the left-hand side is strictly greater than the polynomial interpretation of the right-hand side. E.g. we have:

$$\begin{aligned} P(\mathrm{do}\ x \leftarrow p; (q+r)) &= P(p)^2 \cdot P(q+r)^2 = P(p)^2 \cdot (P(q) + P(r))^2 \\ &> P(p)^2 \cdot P(q)^2 + P(p)^2 \cdot P(r)^2 = P(\mathrm{do}\ x \leftarrow p; q + \mathrm{do}\ x \leftarrow p; r). \end{aligned}$$

The other rule then follows by symmetry.

Let us prove the strong normalisation of the combined relation $\rightarrowtail_{\beta\sigma_1}$ by contradiction: suppose for some simple program t_1 there exists an infinite reduction chain

$$t_1 \rightarrowtail t_2 \rightarrowtail \ldots \tag{2.4}$$

Observe that, by Lemma 2.13, the simplicity of t implies the simplicity of all the t_i. Let $\succ$ be the multiset extension of the strict well-founded partial order $\rightarrowtail_\beta$ (Theorem 1.24). By Lemma 2.9 $\succ$ also must be well-founded. We would like to show that for every i, $\mathfrak{m}(t_i) \succeq \mathfrak{m}(t_{i+1})$, in particular, the inequality becomes strict for the case of β-reduction.

For every i if $t_i \rightarrowtail_{\sigma_1} t_{i+1}$ then $\mathfrak{m}(t_i) = \mathfrak{m}(t_{i+1})$. Indeed, suppose $t_i \doteq C\{p\}$, $t_{i+1} \doteq C\{q\}$ and $p \rightarrowtail q$ is an instance of a σ_1-rule. It directly follows from the definition of $\mathfrak{m}$ that $\mathfrak{m}(p) = \mathfrak{m}(q)$. By Lemma 2.10:

$$\begin{aligned} \mathfrak{m}(t_i) &= \mathfrak{m}(C\{p\}) \\ &= \wr D\{s\} | D \in \mathfrak{m}_1(C), s \in \mathfrak{m}(p) \int \cup\, \mathfrak{m}_0\{C\} \\ &= \wr D\{s\} | D \in \mathfrak{m}(C), s \in \mathfrak{m}(q) \int \cup\, \mathfrak{m}_0(C) \\ &= \mathfrak{m}(C\{q\}) = \mathfrak{m}(t_{i+1}). \end{aligned}$$

On the other hand, if $t_i \rightarrowtail_\beta t_{i+1}$ then $\mathfrak{m}(t_i) \succ \mathfrak{m}(t_{i+1})$. The proof of this fact is more involved. First, consider the case when $t_i \rightarrowtail_\beta t_{i+1}$ is an instance of some β-rule $p \rightarrowtail q$. For most of the β-rules the inequality $\mathfrak{m}(p) \succ \mathfrak{m}(q)$ is immediately seen. The only interesting case is, as usual, the second unit law: $p \doteq (\mathrm{do}\ x \leftarrow \mathrm{ret}\, u; r)$, $q \doteq r[u/x]$. By definition, we have: $\mathfrak{m}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, u; r) = \wr \mathrm{do}\ x \leftarrow \mathrm{ret}\, t; s \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r) \int$ which is strictly greater than $\wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r) \int$. We are left to prove that

$$\wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r) \int \succeq \mathfrak{m}(r[u/x]). \tag{2.5}$$

Let us proceed by induction over the term complexity of r. If $r \doteq x$ or r does not depend on x, the proof is trivial. We have two principal cases to verify.

- r does not have the choice operator on top. Let $r \doteq M\{w\}$ where w is the maximal proper subterm of r containing x, and M is an appropriate context. Note that $\mathfrak{m}(M) = \mathfrak{m}_1(M)$ (hence $\mathfrak{m}_0(M) = \wr\,\S$) and $M[u/x] \doteq M$. Also, observe that r is obtained from w by applying precisely one term-formation rule, and this rule is not choice introduction. We have by Lemma 2.10:

$$\begin{aligned}\mathfrak{m}(r[u/x]) &= \mathfrak{m}(M[u/x]\{w[u/x]\})\\ &= \wr D\{t\} \mid D \in \mathfrak{m}(M), t \in \mathfrak{m}(w[u/x])\S\\ &\preceq \wr D\{s[t/x]\} \mid D \in \mathfrak{m}(M), t \in \mathfrak{m}(u), s \in \mathfrak{m}(w)\S\\ &= \wr (D\{s\})[t/x] \mid t \in \mathfrak{m}(u), D \in \mathfrak{m}(M), s \in \mathfrak{m}(w)\S\\ &= \wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(M\{w\})\S\end{aligned}$$

  and thus the proof of (2.5) is completed.

- $r \doteq r_1 + r_2$ with some programs r_1, r_2. Then the proof of (2.5) runs by Lemma 2.10:

$$\begin{aligned}\mathfrak{m}(r[u/x]) &= \mathfrak{m}(r_1[u/x] + r_2[u/x])\\ &= \mathfrak{m}(r_1[u/x]) \cup \mathfrak{m}(r_2[u/x])\\ &\preceq \wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r_1)\S \cup \wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r_2)\S\\ &= \wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r_1) \cup \mathfrak{m}(r_2)\S\\ &= \wr s[t/x] \mid t \in \mathfrak{m}(u), s \in \mathfrak{m}(r_1 + r_2)\S.\end{aligned}$$

More generally, let $t_i \doteq C\{p\}$, $t_{i+1} \doteq C\{q\}$ where $p \rightarrowtail q$ is a β-rule instance. Then by Lemma 2.10: $\mathfrak{m}(t_i) = \mathfrak{m}(C\{p\}) = \wr D\{s\} \mid D \in \mathfrak{m}_1(C), s \in \mathfrak{m}(p)\S \cup \mathfrak{m}_0(C)$ which is strictly greater than $\wr D\{r\} \mid D \in \mathfrak{m}_1(C), r \in \mathfrak{m}(q)\S \cup \mathfrak{m}_0(C) = \mathfrak{m}(C\{q\})$ because, as we have proved above, $\mathfrak{m}(p) \succ \mathfrak{m}(q)$.

At this point we have constructed an infinite chain of finite multisets of programs $\mathfrak{m}(t_1)$, $\mathfrak{m}(t_2), \ldots$ such that for very i either $\mathfrak{m}(t_i) \succ \mathfrak{m}(t_{i+1})$ or $\mathfrak{m}(t_i) = \mathfrak{m}(t_{i+1})$. If the number of pairs (t_i, t_{i+1}) falling into the former class is infinite we establish a contradiction with the property of $\succ$ of being well-founded. Otherwise, starting from some position of (2.4) onwards, we constantly have $\mathfrak{m}(t_i) = \mathfrak{m}(t_{i+1})$. Therefore, some tail of (2.4) must entirely consist of σ_1-reductions, which contradicts the strong normalisation of $\rightarrowtail_{\sigma_1}$. □

Lemma 2.15. *If $p[x/y] \rightarrowtail_{\beta\sigma_1} q$ for some programs p and q, then there exists a program r such that $p \rightarrowtail_{\beta\sigma_1} r$ and $r[x/y] = q$.*

Proof. Let e.g. $p[x/y] \doteq C\{s\}$, $q \doteq C\{t\}$ where $s \rightarrowtail t$ is a $\beta\sigma_1$-rule instance. It can easily be seen by induction over the term complexity of C that p is of the form $K\{w\}$ where $C \doteq K[x/y]$, $s \doteq w[x/y]$. Observe that all the $\beta\sigma_1$-rules are *left-linear* in the sense of rewriting theory, i.e. the left-hand side of any $\beta\sigma_1$-rule does not contain duplicate metavariables. As a result, $w[x/y] \rightarrowtail t$ must be a specification of some more general $\beta\sigma_1$-rule instance $w \rightarrowtail u$ for which $u[x/y] \doteq t$. Therefore $q \doteq C\{t\} \doteq K[x/y]\{u[x/y]\} \doteq K\{u\}[x/y]$. The proof is now completed by putting $r := K\{u\}$. □

Lemma 2.16. *The reduction relation $\rightarrowtail_{\beta\eta\sigma}$ is confluent and strongly normalising with respect to $\star$-normal programs.*

Proof. We agree that all the programs mentioned in the proof are supposed to be $\star$-normal, without saying this explicitly. Let us denote by $t{\downarrow}^{x}_{\bar{x}}$ the term that is obtained from t by replacing the i-th occurrence of x (if any) by x_i.

Strong normalisation. By Lemma 2.8 it suffices to prove the strong normalisation of $\rightarrowtail_{\beta\sigma_1}$. To that end, we show that every infinite chain of $\beta\sigma_1$-reductions gives rise to an infinite chain of $\beta\sigma_1$-reductions, entirely consisting of simple programs. Therefore the strong normalisability result in question will follow by contradiction with Lemma 2.14. In order to implement the outlined idea let us introduce an auxiliary reduction relation $\rightarrowtail_s$ (carrying out no apparent monad-based interpretation) by the following rules:

$$\begin{array}{rcll}
(\mathrm{do}\ x \leftarrow p; y \leftarrow p; q) & \rightarrowtail & (\mathrm{do}\ x \leftarrow p; q[x/y]) & (x \notin FV(p)) \\
(\mathrm{do}\ x \leftarrow p; y \leftarrow q; r) & \rightarrowtail & (\mathrm{do}\ y \leftarrow q; x \leftarrow p; r) & (x \notin FV(q), y \notin FV(p)) \\
(t+s) & \rightarrowtail & t & \\
(t+s) & \rightarrowtail & s &
\end{array}$$

and a recursive operator ς over programs defined for binding by the assignment:

$$\varsigma(\mathrm{do}\ x \leftarrow p; q) := \mathrm{do}\ x_1 \leftarrow \varsigma(p); \ldots; x_n \leftarrow \varsigma(p); \varsigma(q){\downarrow}^{x}_{\bar{x}}$$

where $n := \max\{1, |q|_x\}$, all the x_i are fresh and by letting ς distribute over the other operations, e.g. $\varsigma(p+q) := \varsigma(p) + \varsigma(q)$. It is easy to see by definition that for any t, $\varsigma(t)$ is a simple program and $\varsigma(t) \rightarrowtail_s^\star t$ (in fact, the first s-rule suffices).

We will be done once we prove that $\rightarrowtail_{\beta\sigma_1}$ quasi-commutes over $\rightarrowtail_s$ (see Definition 1.18). Indeed, suppose we are given an infinite sequence of $\beta\sigma_1$-reductions starting from some program t and establish contradiction. Let us complete the chain at hand by attaching the sequence of s-reductions $\varsigma(t) \rightarrowtail_s^\star t$ on the front. In particular, we obtain thus an infinite reduction:

$$\varsigma(t) \rightarrowtail^\star_{\beta\sigma_1 s} t \rightarrowtail_{\beta\sigma_1} \cdot \rightarrowtail_{\beta\sigma_1} \ldots$$

By Corollary 1.21 we can rearrange the transitions of this chain so as to obtain an arbitrary long prefix of $\beta\sigma_1$ transitions in front. Since the starting term $\varsigma(t)$ is simple, all the following terms reachable by $\beta\sigma_1$ are also simple. Contradiction with Lemma 2.14.

Proving the the quasi-commutation property is predictably laborious and falls into $(4+2)\cdot 4 = 24$ cases of rule combinations. Let us vary over s-reductions.

1. Suppose we have rewritten $w := C\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; s\}$ to $C\{\mathrm{do}\ x \leftarrow t; s[x/y]\}$ by the first s-rule and then rewritten the result by $\rightarrowtail_{\beta\sigma_1}$ once. For the obtained two-step reduction, we need to find an equivalent with the profile $.\rightarrowtail_{\beta\sigma_1} . \rightarrowtail^{\star}_{\beta\sigma_1 s} .$

   If the $\beta\sigma_1$-redex is parallel to the s-contractum, then the reduction in question is trivial. Suppose the $\beta\sigma_1$-redex is a proper subterm of the s-contractum. Then the $\beta\sigma_1$-redex is either a subterm of t or a subterm of $s[x/y]$. In the former case the reduction in question is straightforward; in the latter case it is easily found by Lemma 2.15. Finally, suppose that the $\beta\sigma_1$-redex is a superterm of the s-contractum. In this case we have a bunch of special cases for different mutual arrangements of the s-contractum and the $\beta\sigma_1$-redex depending on the $\beta\sigma_1$-rule applied. We consider only the least straightforward cases of $\beta\sigma_1$-rules: the second unit law and the associativity law. There are three principal opportunities for rewriting by the second unit law. We consider them one by one.

   – Let $C \doteq K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, M; p\}$ and the $\beta\sigma_1$-reduction has from:

   $$K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}; p\} \rightarrowtail K\{p[M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}/z]\}.$$

   Then the reduction in question is as follows:

   $$\begin{aligned} w &\doteq K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, M\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; s\}; p\} \\ &\rightarrowtail_{\beta} K\{p\big[M\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; s\}/z\big]\} \\ &\rightarrowtail^{\star}_{s} K\{p\big[M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}/z\big]\}. \end{aligned}$$

   – Let $C \doteq K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, p; M\}$ and the $\beta\sigma_1$-reduction has from:

   $$K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, p; M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}\} \rightarrowtail K\{M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}[p/z]\}.$$

   Then the reduction in question is as follows:

   $$\begin{aligned} w &\doteq K\{\mathrm{do}\ z \leftarrow \mathrm{ret}\, p; M\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; s\}\} \\ &\rightarrowtail_{\beta} K\{M\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; s\}[p/z]\} \\ &\doteq K\{M[p/z]\{\mathrm{do}\ x \leftarrow t[p/z]; y \leftarrow t[p/z]; s[p/z]\}\} \\ &\rightarrowtail_{s} K\{M[p/z]\{\mathrm{do}\ x \leftarrow t[p/z]; s[p/z][x/y]\}\} \\ &\doteq K\{M\{\mathrm{do}\ x \leftarrow t; s[x/y]\}[p/z]\}. \end{aligned}$$

– Let t be of the form $\mathrm{ret}\, p$ and the $\beta\sigma_1$-reduction is:

$$C\{\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; s[x/y]\} \rightarrowtail C\{s[x/y][p/x]\}.$$

Then the following reduction:

$$\begin{aligned} w \ &\doteq\ C\{\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; y \leftarrow \mathrm{ret}\, p; s\} \\ &\rightarrowtail_\beta C\{\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; s[p/y]\} \\ &\rightarrowtail_\beta C\{s[p/y][p/x]\} \end{aligned}$$

resolves the problem, since evidently $C\{s[p/y][p/x]\} \doteq C\{s[x/y][p/x]\}$.

Concerning the associativity law, if the s-contractum is covered by a term matched against some metavariable from the left-hand side of the associativity law, then the reduction in question is straightforward. Consider the remaining options.

– Let $C \doteq K\{\mathrm{do}\ z \leftarrow \Box; r\}$ and the $\beta\sigma_1$-reduction is:

$$C\{\mathrm{do}\ z \leftarrow (\mathrm{do}\ x \leftarrow t; s[x/y]); r\} \rightarrowtail C\{\mathrm{do}\ x \leftarrow t; z \leftarrow s[x/y]; r\}.$$

Then the reduction in question is as follows:

$$\begin{aligned} w \ &\doteq\ K\{\mathrm{do}\ z \leftarrow (\mathrm{do}\ x \leftarrow t; y \leftarrow t; s); r\} \\ &\rightarrowtail_\beta K\{\mathrm{do}\ x \leftarrow t; z \leftarrow (\mathrm{do}\ y \leftarrow t; s); r\} \\ &\rightarrowtail_\beta K\{\mathrm{do}\ x \leftarrow t; y \leftarrow t; z \leftarrow s; r\} \\ &\rightarrowtail_\beta K\{\mathrm{do}\ x \leftarrow t; (\mathrm{do}\ z \leftarrow s; r)[x/y]\} \\ &\doteq\ K\{\mathrm{do}\ x \leftarrow t; z \leftarrow s; r[x/y]\} \end{aligned}$$

– Let $t \doteq (\mathrm{do}\ z \leftarrow p; q)$ and the $\beta\sigma_1$-reduction has the form:

$$C\big\{\mathrm{do}\ x \leftarrow (\mathrm{do}\ z \leftarrow p; q); s[x/y]\big\} \rightarrowtail C\big\{\mathrm{do}\ z \leftarrow p; x \leftarrow q; s[x/y]\big\}$$

and we are done by the reduction sequence:

$$\begin{aligned} w \ &\doteq\ C\{\mathrm{do}\ x \leftarrow (\mathrm{do}\ z \leftarrow p; q); y \leftarrow (\mathrm{do}\ z \leftarrow p; q); s\} \\ &\rightarrowtail_\beta C\{\mathrm{do}\ x \leftarrow (\mathrm{do}\ z \leftarrow p; q); z \leftarrow p; y \leftarrow q; s\} \\ &\rightarrowtail_s C\{\mathrm{do}\ z \leftarrow p; x \leftarrow (\mathrm{do}\ z \leftarrow p; q); y \leftarrow q; s\} \\ &\rightarrowtail_\beta C\{\mathrm{do}\ z \leftarrow p; v \leftarrow p; x \leftarrow q[v/z]; y \leftarrow q; s\} \\ &\rightarrowtail_s C\{\mathrm{do}\ z \leftarrow p; x \leftarrow q; y \leftarrow q; s\} \\ &\rightarrowtail_s C\{\mathrm{do}\ z \leftarrow p; y \leftarrow q; s[x/y]\}. \end{aligned}$$

Here v is a fresh variable of the same type as z.

2. Suppose we have rewritten $w := C\{\text{do } x \leftarrow t; y \leftarrow s; r\}$ to $C\{\text{do } y \leftarrow s; x \leftarrow t; r\}$ by the second s-rule. We stick to the case of further $\beta\sigma_1$-rewriting by the second unit law and omit verification of the remaining $\beta\sigma_1$-rules. The preliminary analysis from the previous clause perfectly applies here. If the $\beta\sigma_1$-redex is a proper subterm of the s-contractum, the only nontrivial case is as follows: $t \doteq \text{ret}\, p$ and the $\beta\sigma_1$-reduction is:

$$C\{\text{do } y \leftarrow s; x \leftarrow \text{ret}\, p; r\} \rightarrowtail C\{\text{do } y \leftarrow s; r[p/x]\}$$

Then the source two-step reduction can be collapsed to:

$$C\{\text{do } x \leftarrow \text{ret}\, p; y \leftarrow s; r\} \rightarrowtail_\beta C\{\text{do } y \leftarrow s; r[p/x]\}$$

and we are done. Let us consider the remaining case, i.e. the one when the $\beta\sigma_1$-redex is a superterm of the s-contractum. It falls into the following subcases.

– Let $C \doteq K\{\text{do } z \leftarrow \text{ret}\, M; p\}$ and the $\beta\sigma_1$-reduction is

$$K\{\text{do } z \leftarrow \text{ret}\, M\{\text{do } y \leftarrow s; x \leftarrow t; r\}; p\} \rightarrowtail K\{p[M\{\text{do } y \leftarrow s; x \leftarrow t; r\}/z]\}.$$

Then the reduction in question is as follows:

$$\begin{aligned} w \;&\doteq\; K\{\text{do } z \leftarrow \text{ret}\, M\{\text{do } x \leftarrow t; y \leftarrow s; r\}; p\} \\ &\rightarrowtail_\beta K\{p[M\{\text{do } x \leftarrow t; y \leftarrow s; r\}/z]\} \\ &\rightarrowtail_s K\{p[M\{\text{do } y \leftarrow s; x \leftarrow t; r\}/z]\}. \end{aligned}$$

– Let $C \doteq K\{\text{do } z \leftarrow \text{ret}\, p; M\}$ and the $\beta\sigma_1$-reduction is

$$K\{\text{do } z \leftarrow \text{ret}\, p; M\{\text{do } y \leftarrow s; x \leftarrow t; r\}\} \rightarrowtail K\{M\{\text{do } y \leftarrow s; x \leftarrow t; r\}[p/z]\}.$$

Then the reduction in question is as follows:

$$\begin{aligned} w \;&\doteq\; K\{\text{do } z \leftarrow \text{ret}\, p; M\{\text{do } x \leftarrow t; y \leftarrow s; r\}\} \\ &\rightarrowtail_\beta K\{M\{\text{do } x \leftarrow t; y \leftarrow s; r\}[p/z]\} \\ &\doteq\; K\{M[p/z]\{\text{do } x \leftarrow t[p/z]; y \leftarrow s[p/z]; r[p/z]\}\} \\ &\rightarrowtail_s K\{M[p/z]\{\text{do } y \leftarrow s[p/z]; x \leftarrow t[p/z]; r[p/z]\}\} \\ &\doteq\; K\{M\{\text{do } y \leftarrow s; x \leftarrow t; r\}[p/z]\}. \end{aligned}$$

are the reductions in question.

– Let s be of the form $\text{ret}\, p$ with some p, and the $\beta\sigma_1$-reduction is

$$C\{\text{do } y \leftarrow \text{ret}\, p; x \leftarrow t; r\} \rightarrowtail C\{\text{do } x \leftarrow t; r[p/y]\}$$

Then the source two-step reduction can be collapsed to:

$$C\{\text{do } x \leftarrow t; y \leftarrow \text{ret}\, p; r\} \rightarrowtail_\beta C\{\text{do } x \leftarrow t; r[p/y]\}$$

and we are done.

3. Suppose we have rewritten $w := C\{t + s\}$ to $C\{t\}$. If the redex of the following one-step $\beta\sigma_1$-reduction is a subterm of t then the reduction in question is straightforward. Consider the remaining case: the $\beta\sigma_1$-redex is a proper superterm of t. Suppose e.g. the $\beta\sigma_1$-reduction is by the second unit law. We proceed by case analysis.

   – Let $C \doteq K\{\text{do } x \leftarrow \text{ret}\, M; p\}$ and the $\beta\sigma_1$-reduction is:

   $$K\{\text{do } x \leftarrow \text{ret}\, M\{t\}; p\} \rightarrowtail K\{p[M\{t\}/x]\}$$

   and we are done by the reduction:

   $$\begin{aligned} w &\doteq K\{\text{do } x \leftarrow \text{ret}\, M\{t+s\}; p\} \\ &\rightarrowtail_\beta K\{p[M\{t+s\}/x]\} \rightarrowtail_s^\star K\{p[M\{t\}/x]\}. \end{aligned}$$

   – Let $C \doteq K\{\text{do } x \leftarrow \text{ret}\, p; M\}$ and the $\beta\sigma_1$-reduction is:

   $$K\{\text{do } x \leftarrow \text{ret}\, p; M\{t\}\} \rightarrowtail K\{M\{t\}[p/x]\}.$$

   Then we are done by the reduction

   $$\begin{aligned} w &\doteq K\{\text{do } x \leftarrow \text{ret}\, p; M\{t+s\}\} \\ &\rightarrowtail_\beta K\{M\{t+s\}[p/x]\} \doteq K\{M[p/x]\{t[p/x]+s[p/x]\}\} \\ &\rightarrowtail_s K\{M[p/x]\{t[p/x]\}\} \doteq K\{M\{t\}[p/x]\}. \end{aligned}$$

   – Let $C \doteq K\{\text{do } x \leftarrow \Box; p\}$, $t \doteq \text{ret}\, r$ with some r, and the $\beta\sigma_1$-reduction is:

   $$K\{\text{do } x \leftarrow \{\text{ret}\, r\}; p\} \rightarrowtail K\{p[r/x]\}.$$

   Then we are done by the reduction:

   $$\begin{aligned} w &\doteq K\{\text{do } x \leftarrow (\text{ret}\, r + s); p\} \\ &\rightarrowtail_{\sigma_1} K\{\text{do } x \leftarrow \text{ret}\, r; p + \text{do } x \leftarrow s; p\} \\ &\rightarrowtail_s K\{\text{do } x \leftarrow \text{ret}\, r; p\} \\ &\rightarrowtail_\beta K\{p[r/x]\}. \end{aligned}$$

The other $\beta\sigma_1$-reductions are handled analogously.

4. Finally the case when $C\{t+s\}$ rewrites to $C\{s\}$ under the fourth s-rule follows by symmetry from the previous clause.

Confluence. By the strong normalisation property we have just established and Newman's Lemma (cf. e.g. [Klo92]) it suffices to prove local confluence. As $\beta\eta$-reduction is locally confluent by Lemma 1.23 it is only necessary to consider the spans of the form:

$$\cdot\ {}_{\beta\eta}\!\leftarrowtail \cdot \rightarrowtail_{\sigma} \cdot \quad \text{and} \quad \cdot\ {}_{\sigma}\!\leftarrowtail \cdot \rightarrowtail_{\sigma} \cdot$$

The precise analysis is routine and runs in the same way as in Lemma 1.23. □

Theorem 2.17. *The reduction relation $\rightarrowtail_{\beta\eta\sigma\star}$ is confluent and strongly normalising.*

Proof. The proof is the same as the proof of Theorem 1.24 with all the '$\beta\eta$' replaced by the '$\beta\eta\sigma$' and with the call of Lemma 1.23 replaced by the call of Lemma 2.16. □

In the sequel we use the shortening $\mathsf{m} := \beta\eta\sigma\star$. The result of normalisation of term t under m shall be preferably denoted by $\mathsf{nf}(t)$ (rather than the default $\mathsf{nf}_{\mathsf{m}}(t)$). Unless said otherwise, we will use in the sequel terms 'normal', 'normalise', etc. as equivalents of the terms 'm-normal', 'm-normalise', etc.

Recall that $\equiv$ denotes equivalence modulo the ACI-laws.

Theorem 2.18 (Church-Rosser modulo ACI)**.** *For any p and q, $\vdash_{\mathsf{ME}^+} p = q$ iff $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$.*

Proof. Obviously $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$ implies $\vdash_{\mathsf{ME}^+} p = q$. Let us assume that $\vdash_{\mathsf{ME}^+} p = q$ and show the equivalence $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. Let us denote by $\sim$ the symmetric congruent closure of the ACI-laws and observe that by Lemma 1.15, $p\ (\sim \cup \rightarrowtail_{\mathsf{m}} \cup {}_{\mathsf{m}}\!\leftarrowtail)^{\star}\ q$, i.e. there exists n such that

$$w_1 := p \rightharpoondown w_2 \rightharpoondown \ldots \rightharpoondown w_n := q$$

where $\rightharpoondown \in \{\sim, \rightarrowtail_{\mathsf{m}}, {}_{\mathsf{m}}\!\leftarrowtail\}$. Let us show by induction over n that the claim follows from the following statement:

$$\text{for every } s, t \text{ such that } s \sim t,\ \mathsf{nf}(s) \equiv \mathsf{nf}(t). \tag{$*$}$$

If $n = 0$ then $p \doteq q$ and we are trivially done. Let $n > 0$ then $p \rightharpoondown \ldots \rightharpoondown w_{n-1} \rightharpoondown q$. By induction hypothesis, $\mathsf{nf}(p) \equiv \mathsf{nf}(w_{n-1})$. Let us proceed by case distinction. If $w_{n-1} \sim q$ then by ($*$), $\mathsf{nf}(w_n) \equiv \mathsf{nf}(q)$ and thus $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. If $w_{n_1} \rightarrowtail_{\mathsf{m}} q$ or $w_{n_1}\ {}_{\mathsf{m}}\!\leftarrowtail q$ then $\mathsf{nf}(w_n) \equiv \mathsf{nf}(q)$ and again $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. We will be done once we prove ($*$).

If s and t are related either by associativity or by commutativity, ($*$) follows from the more general properties of $\rightarrowtail_{\mathsf{m}}$ called *Church-Rosser modulo associativity* and *Church-Rosser modulo commutativity* which in turn by [JM84] follow from *local commutation* of

$\rightarrowtail_{\mathsf{m}}$ over associativity and commutativity correspondingly. E.g. in case of associativity in order to ensure local commutation, we need to show that whenever t is obtained from s by a one-time application of the associativity law and for some r, $s \rightarrowtail_{\mathsf{m}} r$ then there exist u such that $t \rightarrowtail^{+}_{\mathsf{m}} u$ and r, u are equal up to associativity. The latter can be easily verified by routine analysis of the m-rules one by one.

Unfortunately, $\rightarrowtail_{\mathsf{m}}$ fails to locally commute over idempotence which can be exemplified as follows: $(\mathsf{fst}\langle a,b\rangle + a) \ {}_{\mathsf{m}}\!\leftarrowtail (\mathsf{fst}\langle a,b\rangle + \mathsf{fst}\langle a,b\rangle) \sim \mathsf{fst}\langle a,b\rangle$ but $\mathsf{fst}\langle a,b\rangle$ can not be m-reduced to whatsoever equivalent to $(\mathsf{fst}\langle a,b\rangle + a)$ by idempotence. We thus take another approach. Specifically, we introduce a reduction relation $\rightarrowtail_{\mathsf{i}}$ generated by the rule $u + u \rightarrowtail u$. It is easy to see that $\rightarrowtail_{\mathsf{m}}$ quasi-commutes (see Definition 1.18) over $\rightarrowtail_{\mathsf{i}}$. By Lemma 1.19 the combined relation $\rightarrowtail_{\mathsf{mi}} := \rightarrowtail_{\mathsf{m}} \cup \rightarrowtail_{\mathsf{i}}$ is strongly normalising. By immediate case analysis one can easily establish local confluence of $\rightarrowtail_{\mathsf{mi}}$. By Newman's Lemma (cf. e.g. [Klo92]), we conclude that $\rightarrowtail_{\mathsf{mi}}$ is confluent. In order to prove ($*$), suppose e.g. that s i-reduces to t in one step. Then by confluence, there exists r such that $\mathsf{nf}(s) \rightarrowtail^{\star}_{\mathsf{mi}} r$ and $\mathsf{nf}(t) \rightarrowtail^{\star}_{\mathsf{mi}} r$. Since by Lemma 1.22, $\rightarrowtail^{\star}_{\mathsf{m}}$ commutes over $\rightarrowtail^{\star}_{\mathsf{i}}$, w.l.o.g. $\mathsf{nf}(s) \rightarrowtail^{\star}_{\mathsf{m}} . \rightarrowtail^{\star}_{\mathsf{i}} r$ and $\mathsf{nf}(t) \rightarrowtail^{\star}_{\mathsf{m}} . \rightarrowtail^{\star}_{\mathsf{i}} r$. Note that neither $\mathsf{nf}(s)$ nor $\mathsf{nf}(t)$ can be rewritten under $\rightarrowtail_{\mathsf{m}}$ because both of them are already normal. Therefore, we have: $\mathsf{nf}(s) \rightarrowtail^{\star}_{\mathsf{i}} r \ {}^{\star}_{\mathsf{i}}\!\leftarrowtail \mathsf{nf}(t)$. In particular, $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$. The proof is thus completed. □

Corollary 2.19 (Decidability of ME^{+})**.** *Program equality in* ME^{+} *is decidable.*

Remark 2.20. As we have noted in the introduction, our axiomatisation of nondeterminism is rather strong. One can easily obtain weaker axiomatisation by dropping some axioms. E.g. by dropping the ACI-laws one can obtain an axiomatisation of a list monad. If one only drops the idempotence law, one obtains an axiomatisation that is sound for multiset monads, etc. In a similar way, one can drop some controvertible reduction rules, such as the left distributivity of binding over choice (though this would be inappropriate for process algebra), preservation of $\varnothing$ on the right by binding or perhaps the entire equation about $\varnothing$ altogether. The strong normalisation result we have obtained would imply strong normalisation in all these cases, and the corresponding decidability theorem would be provable using the same argument, without any essential modifications.

Given a data theory $\mathcal{E}$ over a plain signature, we denote by $\equiv_{\mathcal{E}}$ the smallest congruence, encompassing the equations from $\mathcal{E}$ as well as all the instances of ACI-laws.

Theorem 2.21 (Church-Rosser modulo $\equiv_{\mathcal{E}}$)**.** *Let Σ be a plain signature and let $\mathcal{E}$ be a data theory over it. Then for every programs p and q over Σ, $\mathcal{E} \vdash_{\mathsf{ME}^{+}} p = q$ iff $\mathsf{nf}(p) \equiv_{\mathcal{E}} \mathsf{nf}(q)$.*

Proof. Evidently, $\mathsf{nf}(p) \equiv_{\mathcal{E}} \mathsf{nf}(q)$ implies $\mathcal{E} \vdash_{\mathsf{ME}^{+}} p = q$. Let us assume $\mathcal{E} \vdash_{\mathsf{ME}^{+}} p = q$ and prove that $\mathsf{nf}(p) \equiv_{\mathcal{E}} \mathsf{nf}(q)$. Let $\sim$ be the abbreviation for $(\equiv \cup \approx_{\mathcal{E}})$. By definition, $\equiv_{\mathcal{E}}$ is precisely the transitive closure of $\sim$. Observe that by Lemma 1.15, p $(\sim$

$\cup \rightarrowtail_{\mathsf{m}} \cup {}_{\mathsf{m}}\leftarrowtail)^{\star}\ q$, i.e. there exists n such that

$$w_1 := p \rightharpoondown w_2 \rightharpoondown \ldots \rightharpoondown w_n := q$$

where $\rightharpoondown \in \{\sim, \rightarrowtail_{\mathsf{m}}, {}_{\mathsf{m}}\leftarrowtail\}$. In the same way as in Theorem 2.18, we can reduce the problem to the following:

$$\text{for every } s, t \text{ such that } s \sim t,\ \mathsf{nf}(s) \equiv_{\mathcal{E}} \mathsf{nf}(t). \tag{$*$}$$

If s, t in ($*$) are related by one of the ACI-laws, we are done by Theorem 2.18. Assume that for some s, t, the equivalence $s \sim t$ in ($*$) refers to $s \approx_{\mathcal{E}} t$. Observe that by Lemma 1.33, $\approx_{\mathcal{E}}^{*}$ is preserved by $\mathsf{nf}_{\beta\eta\star}$. It is straightforward to see that $\approx_{\mathcal{E}}^{*}$ is also preserved by nf_{σ}. Therefore $\approx_{\mathcal{E}}^{*}$ is preserved by nf_{m} and thus $\mathsf{nf}(s) \approx_{\mathcal{E}}^{*} \mathsf{nf}(t)$. In particular, $\mathsf{nf}(s) \equiv_{\mathcal{E}} \mathsf{nf}(t)$ which completes the proof of ($*$). □

Corollary 2.22. *Given a data theory $\mathcal{E}$ over some plain signature, program equality in* ME^{+} *modulo $\mathcal{E}$ is decidable, provided the conditional word problem in $\mathcal{E}$ is decidable.*

2.3 Contribution and related work

In the present chapter we have introduced an extension of the metalanguage of effects for nondeterminism together with an associated equational calculus ME^{+}. We have justified the notion of a strong additive monad and proved a soundness and completeness theorem. Our notion of an additive monad is rather strong. Essentially, it can be viewed as a further development of the Hasekell's `MonadPlus` type class (cf .e.g. [KSFS05]). An alternative notion of an additive monad is being promoted by Bart Jacobs et al. Starting from the observation that the functorial part of a powerset monad possesses the isomorphisms:

$$\mathcal{P}(X + Y) \cong \mathcal{P}(X) \times \mathcal{P}(Y), \qquad\qquad \mathcal{P}(0) \cong 1$$

one can introduce the analogous laws for any monad and interpret them as the requirement for being additive. When taking this approach, Kleisli hom-sets become naturally enriched very similarly, as required in Definition 2.1. It is remarkable, though, that in this case choice generally fails to be idempotent. Apart from that, Definition 2.1 turns out to be provable as a theorem [CJ10], which shows in particular that in a certain respect our definition of an additive monad is more general. It should also be noted that the definition from [CJ10] involves more categorical structure, e.g. it implies the existence of coproducts. In the view of Theorem 1.37, this does not seem to present an essential difference though.

The most important technical result we have achieved is the strong normalisation theorem (Theorem 2.17). The closest similar result now seems to be strong normalisation of nondeterministic λ-calculus established by de Groote [Gro94]. De Groote's proof is tied to logical derivability in a Gentzen-type system via the Curry-Howard correspondence, and it more or less resembles the classical reducibility argument for the simply typed λ-calculus (cf. e.g. [GTL89]). In spite of the similarity between our reduction system and the reduction system from [Gro94], we did not find any reasonable way to adapt the existing technique and thus developed a completely different proof based on entirely syntactical considerations inspired by the abstract reduction system theory.

As a consequence of the strong normalisability theorem, we obtained a decidability result (Corollary 2.19) for the the calculus of strong additive monads ME^{+}. The latter did not immediately follow from the strong normalisability theorem because the rewriting system did not capture associativity, commutativity and idempotence of nondeterministic choice. In order to to prove decidability we made use of the standard theorems about rewriting modulo a set of equations [JM84] which still did not suffice for the particularly troublesome case of the idempotence law. The latter was therefore handled separately.

Chapter 3

Kleene monads

3.1 Introduction

The metalanguage of effects and its nondeterministic extension from the previous chapter make a suitable background for studying effectful programs. They are powerful enough to reason about finite programs, but they cannot be used for modelling real-world programming languages due to the lack of support of iterative computations. The most straightforward way to introduce a generic recursion principle is by claiming the existence of certain fixed-point operators over hom-sets $\mathsf{Hom}(A, TB)$ and then requiring that certain laws be satisfied, e.g. as it is done in [SP00]. Following this line one can introduce rather rich system of various 'syntactic' notions of recursion and relations between then. On the other hand, there can be a large gap between any such notion of recursion and the real intuition about recursion. The latter can be reasonably formalised by assuming a complete order-enriched structure on $\mathsf{Hom}(A, TB)$ and by imposing some sort of continuity condition over the operators over $\mathsf{Hom}(TA, TA)$. Completeness of a syntactic notion of recursion with respect to an order-enriched interpretation is a property that generally fails, and if it does hold, it usually turns out to be difficult to prove. Still, we believe that even a restricted version of such a completeness theorem is a good argument to make the chosen notion of recursion an appropriate one. And, of course, such a completeness result would immediately produce at least a semi-deciding algorithm for proving program equivalence, which is a fact of crucial importance when it comes to practical implementations. In view of recent works [SP00, HK02b], one could expect such a completeness result in the case under discussion. The principle possibility of such a completeness theorem goes back to the works on iteration theories [BE93].

We take thus the following approach. We introduce the notion of a strong Kleene monad as an extension of the notion of a strong additive monad by a Kleene star operator. We introduce a class of strong continuous Kleene monads to be those strong

Kleene monads which match the order-enriched intuition. Studying the correspondence between the class of all strong Kleene monads and the class of strong continuous Kleene monads is the primary focus for the remainder of this thesis.

3.2 Soundness and completeness

By definition, every Kleisli hom-set $\mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B)$ of an additive monad $\mathbb{T}$ can be considered as a poset $(\mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B), \leqslant)$. The following definition is heavily inspired by Kleene algebra axioms from [Koz94].

Definition 3.1 (Kleene monad)**.** An additive monad $\mathbb{T}$ is a *Kleene monad* if the maps

$$f \mapsto p \oplus f \diamond r \quad \text{and} \quad f \mapsto p \oplus r \diamond f$$

both have least fixed points with respect to $(\mathsf{Hom}(X, TA), \leqslant)$ (recall that $\diamond$ binds more strongly than $\oplus$), and one of the two equivalent conditions holds:

$$\mu f.\,(p \oplus f \diamond r) \diamond q = p \diamond \mu f.\,(q \oplus r \diamond f) \tag{3.1}$$

or

$$\mu f.\,(p \diamond q \oplus f \diamond r) \;=\; p \diamond \mu f.\,(q \oplus f \diamond r), \tag{3.2}$$

$$\mu f.\,(p \diamond q \oplus r \diamond f) \;=\; \mu f.\,(p \oplus r \diamond f) \diamond q. \tag{3.3}$$

A Kleene monad $\mathbb{T}$ is strong if it is a strong additive monad and the strength is continuous in the following sense:

$$\mu f.\,\big(\tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q) \diamond f\big) \;=\; \tau_{A,B}\big(\mathrm{id} \times \mu f.\,(p \oplus q \diamond f)\big)$$

where $p, q \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B)$ and $\mathbf{C}$ is the underlying cartesian category of $\mathbb{T}$.

Let us show that indeed (3.1) is equivalent to the conjunction of (3.2) and (3.3). Assume (3.1). Then $\mu f.(p \diamond q \oplus f \diamond r) \;=\; p \diamond q \diamond \mu f.\,(\eta \oplus r \diamond f) \;=\; p \diamond \mu f.\,(q \oplus f \diamond r)$, hence (3.1) $\implies$ (3.2). By a symmetrical argument, also (3.1) $\implies$ (3.3).

Now assume (3.2) and (3.3). Let us first prove (3.1) with $p = q = \eta$. Since

$$\begin{aligned} &\eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r) \\ &\qquad = \eta \oplus r \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r) \diamond r && \text{[as fix-pt.]} \\ &\qquad = \eta \oplus (\eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r)) \diamond r, && \text{[by def. of } \oplus\text{]} \end{aligned}$$

$\eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r)$ is a fixed point of the map $f \mapsto \eta \oplus f \diamond r$. Hence

$$\eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r) \geqslant \mu f.\,(\eta \oplus f \diamond r).$$

On the other hand, $\mu f.\,(\eta \oplus f \diamond r) \diamond r = r \oplus \mu f.\,(\eta \oplus f \diamond r) \diamond r \diamond r$, which means that $\mu f.\,(\eta \oplus f \diamond r) \diamond r$ is a fixed point of $f \mapsto r \oplus f \diamond r$. Hence $\mu f.\,(\eta \oplus f \diamond r) \diamond r \geqslant \mu f.\,(r \oplus f \diamond r)$. Add η to both sides and obtain

$$\mu f.\,(\eta \oplus f \diamond r) \geqslant \eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r). \tag{3.4}$$

From (3.2) and (3.4), we have $\mu f.(\eta \oplus f \diamond r) = \eta \oplus r \diamond \mu f.\,(\eta \oplus f \diamond r)$, which means that $\mu f.\,(\eta \oplus f \diamond r)$ is a fixed point of $f \mapsto \eta \oplus r \diamond f$, hence $\mu f.(\eta \oplus f \diamond r) \geqslant \mu f.\,(\eta \oplus r \diamond f)$. By the symmetrical argument $\mu f.\,(\eta \oplus r \diamond f) \geqslant \mu f.(\eta \oplus f \diamond r)$. We have thus established

$$\mu f.\,(\eta \oplus f \diamond r) = \mu f.\,(\eta \oplus r \diamond f). \tag{3.5}$$

This implies the general case of (3.1) as follows:

$$\begin{aligned}
&\mu f.\,(p \oplus f \diamond r) \diamond q \\
&\quad = \mu f.\,(p \diamond \eta \oplus f \diamond r) \diamond q && \text{[by 1.2, 1.5]} \\
&\quad = p \diamond \mu f.\,(\eta \oplus f \diamond r) \diamond q && \text{[by 3.2]} \\
&\quad = p \diamond \mu f.\,(\eta \oplus r \diamond f) \diamond q && \text{[by 3.5]} \\
&\quad = p \diamond \mu f.\,(\eta \diamond q \oplus r \diamond f) && \text{[by 3.3]} \\
&\quad = p \diamond \mu f.\,(q \oplus r \diamond f) && \text{[by 1.2, 1.5].}
\end{aligned}$$

This completes the proof of the equivalence (3.1) $\iff$ (3.2) $\wedge$ (3.3). □

The continuity axiom for strength can be weakened a little.

Lemma 3.2. *A strong additive Kleene monad $\mathbb{T}$ is a strong Kleene monad iff the inequation*

$$\mu f.\,(\tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q) \diamond f) \geqslant \tau_{A,B}(\mathrm{id} \times \mu f.\,(p \oplus q \diamond f))$$

holds for every $p, q \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B)$ and $\mathbf{C}$ is the underlying cartesian category of $\mathbb{T}$.

Proof. By definition, we need to show that the opposite inequality holds over all strong additive Kleene monads. We will be done once we prove that $\tau_{A,B}(\mathrm{id} \times \mu f.\,(p \oplus q \diamond f))$ is a fixed point of the map $f \mapsto \tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q) \diamond f$. Indeed, we have:

$$\begin{aligned}
&\tau_{A,B}(\,\mathrm{id} \times \mu f.\,(p \oplus q \diamond f)) \\
&\quad = \tau_{A,B}(\mathrm{id} \times (p \oplus q \diamond \mu f.\,(p \oplus q \diamond f))) && \text{[as fix-pt.]} \\
&\quad = \tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q \diamond \mu f.\,(p \oplus q \diamond f))) && \text{[by 2.2]} \\
&\quad = \tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q) \diamond \tau_{A,B}(\mathrm{id} \times \mu f.\,(p \oplus q \diamond f)) && \text{[by 1.9]}
\end{aligned}$$

and thus the proof is complete. □

Equation (3.1) is a minimal requirement for the Kleene star operator to be defined correctly. But typically, one deals with Kleene monads satisfying the somewhat more restrictive (but natural) condition of *ω-continuity*.

Definition 3.3 (ω-continuous Kleene monad)**.** A (strong) Kleene monad is *ω-continuous* if for every p, q and r, $\mu f.\,(p \oplus f \diamond r) \diamond q$ $(= p \diamond \mu f.\,(q \oplus r \diamond f))$ is the least upper bound of the family of morphisms having form $p \diamond r \diamond \ldots \diamond r \diamond q$. In particular, if $p = \eta$ then $\mu f.\,(q \oplus r \diamond f)$ is the supremum of $\{r \diamond \ldots \diamond r \diamond q\}$ and if $q = \eta$ then $\mu f.\,(p \oplus f \diamond r)$ is the supremum of $\{p \diamond r \diamond \ldots \diamond r\}$.

Note that we do not introduce any additional axioms for strength continuity specific for ω-continuity. Since we are not going to introduce any further notion of continuity, the prefix 'ω-' shall be commonly omitted in the sequel.

All Kleene monads appearing naturally are continuous. Like in the case of additive monads, the functorial parts of Kleene monads typically involve powerset. In particular, the powerset monad $\mathcal{P}$ and the countable powerset monad $\mathcal{P}_\omega$ can easily be seen as strong continuous Kleene monads, unlike the finite powerset monad $\mathcal{P}_{fin}$, which is a strong additive monad but not a Kleene monad.

One can view the powerset functor $\mathcal{P}$ as $\lambda X.\, 2^X$, which suggests the following generalisation: by replacing 2 with Q where Q is a *quantale* object, i.e. an object carrying the structure of a complete lattice with non-commutative multiplication [Ros90], from which we obtain a *quantale* monad (cf. e.g [Jac10]). Every quantale monad is indeed a strong continuous Kleene monad. Sensible instantiations of Q besides $Q := 2$ include $Q := 3 = 1 + 1 + 1$, viewed as a domain of a three-valued logic and $Q := [0, 1]$, i.e. the real-valued interval with the standard complete lattice structure and the quantale multiplication put by definition equal to join. Quantales are commonly considered as standard models of Kleene algebras. The relation between Kleene algebras and quantales is thus somewhat similar to the relation between continuous Kleene monads and Kleene monads, but notably $\lambda X.\, Q^X$ does not make a Kleene monad if Q is only required to be a Kleene algebra, and actually in this case $\lambda X.\, Q^X$ even fails to be a monad.

We would like to outline the class of strong (continuous) Kleene monads by proving an analogue of Theorem 2.5. In order to simplify the calculations we introduce one more class of monads with good modularity properties and still of interest in itself.

Definition 3.4 (Completely additive monads)**.** A *completely additive monad* is a monad whose Kleisli category is enriched over complete lattices. A *strong completely additive monad* is a completely additive monad that is strong and such that for all objects A, B of the underlying category $\Delta_{A,B}$ is continuous, i.e. for every appropriate set of morphisms S, $\Delta_{A,B}(\sup S) = \sup\{\Delta_{A,B}(f) \mid f \in S\}$.

Note that by definition, every (strong) completely additive monad is (strong) additive. Moreover, given an additive monad, in order to establish completeness it suffices to ensure that the partial order of every Kleisli hom-set is complete. Indeed, by definition, every Kleisli hom-set of an additive monad possesses the structure of a bounded join-semilattice. Hence if the underlying partial order is complete, then the lub of any set S can be defined as the lub of the directed set of all finite joins of elements from S, and thus every Kleisli hom-set becomes a complete join-semilattice, i.e. a complete lattice.

Lemma 3.5. *Every (strong) completely additive monad is a (strong) continuous Kleene monad.*

Proof. Let $\mathbb{T}$ be a completely additive monad over a category $\mathbf{C}$. Then Kleisli composition must be continuous in both arguments. Since $\oplus$ is obviously continuous, then both the mappings $f \mapsto p \oplus f \diamond r$ and $f \mapsto q \oplus r \diamond f$ must be continuous as compositions of continuous functions. By Kleene's fixed-point theorem, both of them must possess least fixed points. Moreover, $\mu f.\,(p \oplus f \diamond r)$ must be the supremum of the set $\{p, p \diamond r, p \diamond r \diamond r, \ldots\}$ and $\mu f.\,(q \oplus r \diamond f)$ must be the supremum of the set $\{q, r \diamond q, r \diamond r \diamond q, \ldots\}$. The equation (3.1) now follows by the continuity of Kleisli composition. By definition, $\mathbb{T}$ is a continuous Kleene monad.

Suppose $\mathbb{T}$ is moreover a strong additive monad, and prove that $\mathbb{T}$ is a strong continuous Kleene monad. We are left to prove strength continuity. Let $A, B \in \mathrm{Ob}\,(\mathbf{C})$ and let $p, q \in \mathsf{Hom}_{\mathbf{C}_{\mathbb{T}}}(A, B)$. Then

$$\begin{aligned}
&\mu f.\,(\tau_{A,B}(\mathrm{id} \times p) \oplus \tau_{A,B}(\mathrm{id} \times q) \diamond f)\\
&\qquad = \sup\{\tau_{A,B}(\mathrm{id} \times p), \tau_{A,B}(\mathrm{id} \times q) \diamond \tau_{A,B}(\mathrm{id} \times p), \ldots\} && \text{[by def.]}\\
&\qquad = \sup\{\tau_{A,B}(\mathrm{id} \times p), \tau_{A,B}(\mathrm{id} \times q \diamond p), \ldots\} && \text{[by 1.9]}\\
&\qquad = \sup\{\Delta_{A,B}(p), \Delta_{A,B}(q \diamond p), \ldots\} && \text{[by 2.3]}\\
&\qquad = \Delta_{A,B}(\sup\{p, q \diamond p, \ldots\}) && \text{[by cont.]}\\
&\qquad = \tau_{A,B}(\mathrm{id} \times \sup\{p, q \diamond p, \ldots\}) && \text{[by 2.3]}\\
&\qquad = \tau_{A,B}(\mathrm{id} \times \mu f.\,(p \oplus q \diamond f)). && \text{[by def.]}
\end{aligned}$$

Therefore $\mathbb{T}$ is a strong continuous Kleene monad, and we are done. □

It can easily be seen that given a quantale object Q, $\lambda X.\,Q^X$ is a strong completely additive monad. Using it as a basis, we can generate further strong completely additive monads (and hence, by Lemma 3.5, strong continuous Kleene monads) by the following

Theorem 3.6. *Given a strong completely additive monad $\mathbb{T}$, the strong additive monads obtained from it by applying state, input, output and continuation monad transformers are also strong completely additive.*

Proof. The argument is the same as the one used in the proof of Theorem 2.5. It is clear that the constructions provided in that theorem are robust enough to bear generalisation from finite joins and monotone functions to whatever joins and continuous functions. □

Although the conditions of Definition 3.1 look rather mild, they conceal many properties that may not be evident at first sight. It is more convenient to formulate them in the internal language of Kleene monads, which will be introduced later. But some of these properties are necessary in order to establish the correctness of this language.

Lemma 3.7. *Provided $b \diamond a = a \diamond c$ for some appropriate morphisms a, b and c,*

$$\mu f.\,(a \oplus b \diamond f) = \mu f.\,(a \oplus f \diamond c)$$

Proof. The proof is by establishing mutual inequality. For one thing we have:

$$\begin{aligned}
& a \oplus \mu f.\,(a \oplus b \diamond f) \diamond c \\
&\quad = a \oplus \mu f.\,(a \diamond c \oplus b \diamond f) && \text{[by 3.3]} \\
&\quad = a \oplus \mu f.\,(b \diamond a \oplus b \diamond f) && \text{[by assum.]} \\
&\quad = a \oplus \mu f.\,(\eta \oplus b \diamond f) \diamond b \diamond a && \text{[by 3.3]} \\
&\quad = (\eta \oplus \mu f.\,(\eta \oplus b \diamond f) \diamond b) \diamond a && \text{[by distr.]} \\
&\quad = (\eta \oplus \mu f.\,(\eta \oplus f \diamond b) \diamond b) \diamond a && \text{[by 3.1]} \\
&\quad = \mu f.\,(\eta \oplus f \diamond b) \diamond a && \text{[as fix-pt.]} \\
&\quad = \mu f.\,(a \oplus b \diamond f). && \text{[by 3.3]}
\end{aligned}$$

Therefore, the morphism $\mu f.\,(a \oplus b \diamond f)$ must be a fixed point of $f \mapsto a \oplus f \diamond c$, hence $\mu f.\,(a \oplus b \diamond f) \geqslant \mu f.\,(a \oplus f \diamond c)$. By symmetry, $\mu f.\,(a \oplus b \diamond f) \leqslant \mu f.\,(a \oplus f \diamond c)$ and thus we are done. □

Lemma 3.8. *Let $\mathbb{T}$ be a strong Kleene monad over a cartesian category $\mathbf{C}$. Then, given two Kleisli morphisms $a, b \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B)$, the fixed point*

$$t := \mu f.\,(\tau_{A,B}\langle \pi_1, a\rangle \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f)$$

satisfies the identity: $t = \tau_{A,B}\langle \pi_1, T\pi_2 \circ t\rangle$.

Proof. Let $s := \mu f.\,\big(\tau_{A,A\times B}(\mathrm{id} \times \tau_{A,B}\langle \pi_1, a\rangle) \oplus \tau_{A,A\times B}(\mathrm{id} \times \tau_{A,B}\langle \pi_1, b\rangle) \diamond f\big)$. By continuity of the strength:

$$s = \tau_{A,A\times B}(\mathrm{id} \times t). \tag{3.6}$$

In order to prove the identity of t in question, it suffices to prove the following auxiliary identity of s:

$$\eta\langle \pi_1\pi_2, \pi_2\rangle \diamond s = s \diamond \eta\langle \pi_1\pi_2, \pi_2\rangle. \tag{3.7}$$

Indeed, (3.7) implies the goal as follows:

$$\eta\langle \pi_1\pi_2, \pi_2\rangle \diamond s = s \diamond \eta\langle \pi_1\pi_2, \pi_2\rangle$$

$$\begin{aligned}
&\iff \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times t) \\
&\qquad = \tau_{A,A\times B}(\mathrm{id}\times t) \diamond \eta\langle\pi_1\pi_2,\pi_2\rangle && \text{[by 3.6, 1.5, 1.2]} \\
&\qquad = \tau_{A,A\times B}(\mathrm{id}\times t) \circ \langle\pi_1\pi_2,\pi_2\rangle \\
&\iff \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times t) \\
&\qquad = \tau_{A,A\times B}(\pi_1\pi_2, t\circ\pi_2) \\
&\implies \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times t) \diamond \eta\langle\pi_1,\mathrm{id}\rangle \\
&\qquad = \tau_{A,A\times B}(\pi_1\pi_2, t\circ\pi_2) \diamond \eta\langle\pi_1,\mathrm{id}\rangle \\
&\iff \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times t) \circ \langle\pi_1,\mathrm{id}\rangle && \text{[by 1.5, 1.2]} \\
&\qquad = \tau_{A,A\times B}(\pi_1\pi_2, t\circ\pi_2) \circ \langle\pi_1,\mathrm{id}\rangle \\
&\iff \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\pi_1\times t) = \tau_{A,A\times B}(\pi_1, t) \\
&\implies \eta(\mathrm{id}\times\pi_2) \diamond \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\pi_1\times t) \\
&\qquad = \eta(\mathrm{id}\times\pi_2) \diamond \tau_{A,A\times B}\langle\pi_1, t\rangle \\
&\iff \eta(\mathrm{id}\times\pi_2) \circ \langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\pi_1\times t) && \text{[by 1.5, 1.2, 1.3]} \\
&\qquad = T(\mathrm{id}\times\pi_2) \circ \tau_{A,A\times B}\langle\pi_1, t\rangle \\
&\iff \eta\langle\pi_1\pi_2,\pi_2\pi_2\rangle \diamond \tau_{A,A\times B}(\pi_1\times t) && \text{[by nat. of } \tau] \\
&\qquad = \tau_{A,B}\langle\pi_1, T\pi_2\circ t\rangle \\
&\iff T\langle\pi_1\pi_2,\pi_2\pi_2\rangle \circ \tau_{A,A\times B}(\pi_1\times t) && \text{[by 1.5, 1.3]} \\
&\qquad = \tau_{A,B}\langle\pi_1, T\pi_2\circ t\rangle \\
&\iff T\pi_2 \circ \tau_{A,A\times B}(\pi_1\times t) = \tau_{A,B}\langle\pi_1, T\pi_2\circ t\rangle \\
&\iff t = \tau_{A,B}(\pi_1, T\pi_2\circ t) && \text{[by 1.2]}
\end{aligned}$$

We are left to prove (3.7). First, observe the following commutation property, which is an elementary consequence of (1.2), (1.3) and (1.5).

$$\begin{aligned}
&\eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,c\rangle) \\
&\qquad\qquad = \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,c\rangle) \diamond \eta\langle\pi_1\pi_2,\pi_2\rangle,
\end{aligned} \tag{3.8}$$

where $c \in \{a, b\}$. Therefore we have:

$$\begin{aligned}
&\eta\langle\pi_1\pi_2,\pi_2\rangle \diamond s \\
&\quad = \eta\langle\pi_1\pi_2,\pi_2\rangle \diamond \mu f.\,(\tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,a\rangle) \oplus \\
&\qquad \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,b\rangle) \diamond f) && \text{[by def. of } s] \\
&\quad = \mu f.\,(\eta\langle\pi_1\pi_2,\pi_2\rangle \oplus f \diamond \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,b\rangle)) \diamond \\
&\qquad \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,a\rangle) && \text{[by 3.1]} \\
&\quad = \mu f.\,(\eta\langle\pi_1\pi_2,\pi_2\rangle \oplus \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,b\rangle) \diamond f) \diamond && \text{[by 3.8,} \\
&\qquad \tau_{A,A\times B}(\mathrm{id}\times\tau_{A,B}\langle\pi_1,a\rangle) && \text{Lemma 3.7]}
\end{aligned}$$

$$
\begin{aligned}
&= \mu f.\,(\eta\langle \pi_1\pi_2, \pi_2\rangle \diamond \tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, a\rangle) \oplus \\
&\quad \tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, b\rangle) \diamond f) && \text{[by 3.3]}\\
&= \mu f.\,(\tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, a\rangle) \diamond \eta\langle \pi_1\pi_2, \pi_2\rangle \oplus \\
&\quad \tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, b\rangle) \diamond f) && \text{[by 3.8]}\\
&= \mu f.\,(\tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, a\rangle) \oplus \\
&\quad \tau_{A,A\times B}(\mathrm{id}\times \tau_{A,B}\langle \pi_1, b\rangle) \diamond f) \diamond \eta\langle \pi_1\pi_2, \pi_2\rangle && \text{[by 3.3]}\\
&= s \diamond \eta\langle \pi_1\pi_2, \pi_2\rangle && \text{[by def. of } s\text{]}
\end{aligned}
$$

and thus the whole claim is proved. □

Corollary 3.9. *Given two morphisms $a, b \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(A, B)$, the fixed point*

$$t := \mu f.\,(\tau_{A,B}\langle \mathrm{id}, a\rangle \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f)$$

satisfies the identity: $t = \tau_{A,B}\langle \mathrm{id}, T\pi_2 \circ t\rangle$.

Proof. Note that

$$
\begin{aligned}
t\circ\pi_1 &= \mu f.\,(\tau_{A,B}\langle \mathrm{id}, a\rangle \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f) \diamond \eta \circ \pi_1 && \text{[by 1.2, 1.5]}\\
&= \mu f.\,(\tau_{A,B}\langle \mathrm{id}, a\rangle \diamond \eta\circ\pi_1 \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f) && \text{[by 3.3]}\\
&= \mu f.\,(\tau_{A,B}\langle \mathrm{id}, a\rangle \circ\pi_1 \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f) && \text{[by 1.2, 1.5]}\\
&= \mu f.\,(\tau_{A,B}\langle \pi_1, a\circ\pi_1\rangle \oplus \tau_{A,B}\langle \pi_1, b\rangle \diamond f)
\end{aligned}
$$

Therefore, Lemma 3.8 under the assignments $a := a\circ\pi_1$ and $b := b$ results in

$$t\circ\pi_1 = \tau_{A,B}\langle \pi_1, T\pi_2\circ t\circ\pi_1\rangle = \tau_{A,B}\langle \mathrm{id}, T\pi_2\circ t\rangle\circ\pi_1,$$

which implies the identity in question. □

The relation between the continuity axiom for strength and the identity of Lemma 3.8 can be illustrated by the following diagrams.

$$\begin{tikzcd}
A \arrow[d, "\mathrm{id}"'] \arrow[ddddd, bend right=60, "\mathrm{id}"'] & \times & B \arrow[d, "a"'] \\
A \arrow[d, "\mathrm{id}"'] & \times & B \arrow[d, "b"'] \\
A \arrow[d, "\mathrm{id}"'] & \times & B \arrow[d, "b"'] \\
A & \times & B \\
\vdots & & \vdots \\
A & \times & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[d, "\mathrm{id}"'] \arrow[drr, "a"] \arrow[ddddd, bend right=60, "\mathrm{id}"'] & \times & B \arrow[d] \\
A \arrow[d, "\mathrm{id}"'] \arrow[drr, "b"] & \times & B \arrow[d] \\
A \arrow[d, "\mathrm{id}"'] \arrow[drr, "b"] & \times & B \arrow[d] \\
A & \times & B \\
\vdots & & \vdots \\
A & \times & B
\end{tikzcd}$$

The left diagram schematically depicts the following fact: if a program variable is neither read nor updated in a loop it remains the same as it was initially. This is precisely the intended meaning of strength continuity. The right diagram shows moreover that the value of the variable also remains unchanged if it is read but still never updated. Roughly speaking, Lemma 3.8 states that the left diagram implies the right one. There is a notable similarity between these diagrams and the diagrams from [EL00] illustrating one of the axioms for recursive binding. The exact correspondence between our axioms and the axioms for recursive binding from [EL00] is nevertheless far from being lucid, in particular because the latter concept is formulated in terms of partial functions, whereas Kleene monads do not provide facilities to reason about the partiality of primitive morphisms.

The calculations performed in the few recent proofs were written in a low-level language, simulating the calculus of commutative diagrams. Fortunately, the reasoning about Kleene monads can be considerably facilitated by using an extension of the metalanguage of effects by the following *Kleene star* term constructor.

$$\frac{\Gamma \rhd p : TA \quad \Gamma, x : A \rhd q : TA}{\Gamma \rhd \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star : TA}$$

Intuitively the $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star)$ construct denotes a nondeterministically chosen number of iterated executions of q, initialised by $x \leftarrow p$, with the result x of the computation fed through the loop. The metalanguage of effects, extended by the Kleene star, shall be called the *metalanguage of control and effects (MCE)*.

For an arbitrary natural n we shall also use the notation $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^n)$ referring to p if $n = 0$ and to $\big(\mathsf{init}\, x \leftarrow (\mathsf{do}\ \ x \leftarrow p; q) \,\mathsf{in}\, q^{n-1}\big)$ if $n > 0$. Also, we adopt for the Kleene star the same convention as for binding, e.g. given a program p whose return type is $T(A_1 \times \ldots \times A_n)$ we use the notation $(\mathsf{init}\, \bar{z} \leftarrow p \,\mathsf{in}\, q^\star)$ as a shortening for

$$\mathsf{init}\, z \leftarrow p \,\mathsf{in}\, q[\mathrm{pr}_1^n(z)/z_1, \ldots, \mathrm{pr}_n^n(z)/z_n]^\star.$$

where z is an appropriate fresh variable. Programs not containing a Kleene star will be referred to as *non-iterative*. Another convenient shortening is $(\mathsf{init}\,\bar{x} \leftarrow p\,\mathsf{in}\,q^\star\,\mathsf{res}\,\bar{y} \to r)$ for $(\mathsf{do}\ \bar{y} \leftarrow (\mathsf{init}\,\bar{x} \leftarrow p\,\mathsf{in}\,q^\star); r)$. Similarly, $(\mathsf{init}\,\bar{x} \leftarrow p\,\mathsf{in}\,q^n\,\mathsf{res}\,\bar{y} \to r)$ will refer to $(\mathsf{do}\ \bar{y} \leftarrow (\mathsf{init}\,\bar{x} \leftarrow p\,\mathsf{in}\,q^n); r)$.

The formal interpretation of a Kleene star is given as follows:

$$[\![\Gamma \rhd \mathsf{init}\,x \leftarrow p\,\mathsf{in}\,q^\star : TA]\!] := T\pi_2 \circ \mu f.\,(\tau_{[\![\Gamma]\!],[\![A]\!]}\langle id, h\rangle \oplus \tau_{[\![\Gamma]\!],[\![A]\!]}\langle \pi_1, g\rangle \diamond f),$$

where $h := [\![\Gamma \rhd p : TA]\!]$ and $g := [\![\Gamma, x : A \rhd q : TA]\!]$. This definition is not the most intuitive one, but it is taken as the basis because the fixed point involved exists by definition. We will see right away that an equivalent definition can be given but it requires an additional argument ensuring the existence of a fixed point involved.

Lemma 3.10. *In any strong Kleene monad $\mathbb{T}$, for any Kleisli morphisms $h \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(C, D)$ and $g \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(C \times D, D)$, the endomap*

$$f \mapsto \tau_{C,D}\langle \mathrm{id}, h \oplus g \diamond f\rangle \tag{3.9}$$

over $f \in \mathsf{Hom}_{\mathbf{C}^{\mathbb{T}}}(C, C \times D)$ possesses the least fixed point. Moreover, if $C = [\![\Gamma]\!]$, $D = [\![A]\!]$ and $h = [\![\Gamma \rhd p : TA]\!]$, $g = [\![\Gamma, x : A \rhd q : TA]\!]$ with arbitrary well-defined expressions in brackets, the interpretation of $(\Gamma \rhd \mathsf{init}\,x \leftarrow p\,\mathsf{in}\,q^\star : TA)$ can be equivalently given by the expression

$$T\pi_2 \circ \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h \oplus g \diamond f\rangle)$$

Proof. Since the least fixed point, if it exists, is unique, we will be done once we show that the fixed point

$$\mu f.\,(\tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) \tag{3.10}$$

is the least fixed point of (3.9). This proof consists of two parts: (i) the morphism (3.10) is invariant under (3.9) and (ii) every r, invariant under (3.9), is greater than (3.10).

Prove (i). Let us refer to the fixed point (3.10) as s. Then

$$\begin{aligned}
s &= \tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond s \\
&= \tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond \tau_{C,D}\langle \mathrm{id}, T\pi_2 \circ s\rangle && \text{[by Cor. 3.9]} \\
&= \tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \mathrm{id}, g \diamond \tau_{C,D}\langle \mathrm{id}, T\pi_2 \circ s\rangle\rangle && \text{[by 1.8]} \\
&= \tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \mathrm{id}, g \diamond s\rangle && \text{[by Cor. 3.9]}
\end{aligned}$$

and thus s is indeed invariant under (3.9).

Prove (ii). Suppose $r = \tau_{C,D}\langle \mathrm{id}, h \oplus g \diamond r\rangle$. By (2.2) $r = \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \mathrm{id}, g \diamond r\rangle$. Then

$$t_{C,C\times D}\langle \mathrm{id}, r\rangle$$

$$
\begin{aligned}
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \mathrm{id}, g \diamond r\rangle\rangle \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, g \diamond r\rangle\rangle && \text{[by 2.2]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, g \diamond T\pi_2 \circ \tau_{C,C\times D}\langle \mathrm{id}, r\rangle\rangle\rangle && \text{[by 1.6]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, g^\dagger \circ (\eta\pi_2)^\dagger \circ \tau_{C,C\times D}\langle \mathrm{id}, r\rangle\rangle\rangle && \text{[by 1.5, 1.3]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, (g \circ \pi_2)^\dagger \circ \tau_{C,C\times D}\langle \mathrm{id}, r\rangle\rangle\rangle && \text{[by 1.2]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, (g \circ \pi_2) \diamond \tau_{C,C\times D}\langle \mathrm{id}, r\rangle\rangle\rangle && \text{[by 1.5]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}(\mathrm{id} \times g) \diamond \tau_{C,C\times D}\langle \mathrm{id}, r\rangle\rangle && \text{[by 1.8]} \\
&= \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \\
&\quad \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond \tau_{C,C\times D}\langle \mathrm{id}, r\rangle && \text{[by 1.8]}
\end{aligned}
$$

which means that $\tau_{C,C\times D}\langle \mathrm{id}, r\rangle$ is a fixed point of the map

$$f \mapsto \tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond f.$$

Therefore by strength continuity and (3.3):

$$
\begin{aligned}
&\tau_{C,C\times D}\langle \mathrm{id}, r\rangle \\
&\quad \geqslant \mu f.\,(\tau_{C,C\times D}\langle \mathrm{id}, \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle \oplus \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond f) \\
&\quad = \mu f.\,(\tau_{C,C\times D}\langle \pi_1, \eta\rangle \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond f) \\
&\quad = \mu f.\,(\tau_{C,C\times D}\langle \pi_1, \eta\rangle \oplus \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond f) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle.
\end{aligned}
$$

The identity $\tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond \tau_{C,C\times D}\langle \pi_1, \eta\rangle = \tau_{C,C\times D}\langle \pi_1, \eta\rangle \diamond \tau_{C,D}\langle \pi_1, g\rangle$, elementary following from (1.8), allows us to continue the calculation by calling Lemma 3.7 with $a := \tau_{C,C\times D}\langle \pi_1, \eta\rangle$, $b := \tau_{C,D}\langle \pi_1, g\rangle$ and $c := \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle$ as follows

$$
\begin{aligned}
&\mu f.\,(\tau_{C,C\times D}\langle \pi_1, \eta\rangle \oplus \tau_{C,C\times D}\langle \pi_1, \tau_{C,D}(\mathrm{id} \times g)\rangle \diamond f) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \\
&\quad = \mu f.\,(\tau_{C,C\times D}\langle \pi_1, \eta\rangle \oplus f \diamond \tau_{C,D}\langle \pi_1, g\rangle) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \\
&\quad = \tau_{C,C\times D}\langle \pi_1, \eta\rangle \diamond \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)
\end{aligned}
$$

In summary, we have proved

$$\tau_{C,C\times D}\langle \mathrm{id}, r\rangle \geqslant \tau_{C,C\times D}\langle \pi_1, \eta\rangle \diamond \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)$$

The proof can be completed by composing both sides of the latter inequality with $T\pi_2$ on the left and further simplification of the result by (1.6). □

(unf₁) $\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star = p + \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \,\mathsf{res}\, x \to q$

(unf₂) $\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star = p + \mathsf{init}\, x \leftarrow (\mathsf{do}\ x \leftarrow p; q;) \,\mathsf{in}\, q^\star$

(init) $\mathsf{init}\, x \leftarrow (\mathsf{do}\ y \leftarrow p; q) \,\mathsf{in}\, r^\star = \mathsf{do}\ y \leftarrow p; (\mathsf{init}\, x \leftarrow q \,\mathsf{in}\, r^\star)$

(ind₁) $\dfrac{\mathsf{do}\ x \leftarrow p; q \leqslant p}{\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \leqslant p}$ **(ind₂)** $\dfrac{\mathsf{do}\ x \leftarrow q[y/x]; r \leqslant r[y/x]}{\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \,\mathsf{res}\, x \to r \leqslant \mathsf{do}\ x \leftarrow p; r}$

Here r in **(init)** *and* **(ind₂)** *satisfies the condition:* $y \notin \mathrm{Vars}(r)$.

FIGURE 3.1: Axioms and rules for Kleene monads

The calculus of Kleene monads $\mathsf{ME}^\star$ extends the calculus of additive monads ME^+ (Fig. 2.1) by the additional axioms and rules presented in Fig. 3.1 together with the evident congruence rule **(cong_star)**. As usual, we leave the variable contexts implicit. The symbol $\leqslant$ appearing in **(ind₁)** and **(ind₂)** is merely an abbreviation: for every p, q, $p \leqslant q$ is supposed to be read as $p + q = q$.

Remark 3.11. The rule **(ind₂)** looks the most complicated, particularly because of the substitution operator involved. The reason for it is purely technical: we need it in order to match the variable contexts of both sides. In more detail, observe that the term-formation rules do not allow for constructing terms like $(\mathsf{do}\ x \leftarrow q; r)$ with q, depending on x. Therefore to balance the variable contexts, the x variable (possibly) occurring in r on the right-hand side of **(ind₂)** must be shadowed. The condition $y \notin \mathrm{Vars}(r)$ ensures that the described situation is indeed avoided.

Theorem 3.12 (Soundness and completeness)**.** *The calculus* $\mathsf{ME}^\star$ *is sound and strong complete over strong Kleene monads.*

Proof. In the scope of this proof, let $C := [\![\Gamma]\!]$, $D := [\![A]\!]$ and $E := [\![B]\!]$.

Soundness. In the view of Theorem 2.7 it suffices to prove only the soundness of the three new axioms and the two new rules presented in Fig. 3.1. We check them one by one.

Axiom **(unf₁)**. By definition, $[\![\Gamma \rhd \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star : TA]\!] = T\pi_2 \circ t$ where t is the fixed point $\mu f.\, (\tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)$, $h := [\![\Gamma \rhd p : TA]\!]$ and $g := [\![\Gamma, x : A \rhd q : TA]\!]$. Then, by definition of the least fixed point and by (1.6):

$$\begin{aligned} T\pi_2 \circ t &= T\pi_2 \circ \mu f.\, (\tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond t) \\ &= h \oplus g \diamond \mu f.\, (\tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) \end{aligned}$$

By Corollary 3.9, the latter is equivalent to $h \oplus g \diamond \tau_{C,D}\langle \mathrm{id}, T\pi_2 \circ t\rangle$ which is precisely the interpretation of $p + \mathsf{do}\ x \leftarrow (\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star); q$.

Axiom **(unf$_2$)**. Let $[\![\Gamma \rhd \mathrm{init}\, x \leftarrow p \,\mathrm{in}\, q^\star : TA]\!]$ be as in the previous clause. We transform $T\pi_2 \circ t$ as follows:

$$\begin{aligned}
T\pi_2 \circ t &= T\pi_2 \circ \mu f.\,(\tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)\\
&= T\pi_2 \circ (\mu f.\,(\eta \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) \diamond \tau_{C,D}\langle id, h\rangle) && \text{[by 3.3]}\\
&= T\pi_2 \circ (\mu f.\,(\eta \oplus f \diamond \tau_{C,D}\langle \pi_1, g\rangle) \diamond \tau_{C,D}\langle id, h\rangle) && \text{[by 3.1]}\\
&= T\pi_2 \circ (\eta \diamond \tau_{C,D}\langle id, h\rangle \oplus \mu f.\,(\eta \oplus f \diamond \tau_{C,D}\langle \pi_1, g\rangle) \diamond\\
&\quad \tau_{C,D}\langle \pi_1, g\rangle \diamond \tau_{C,D}\langle id, h\rangle) && \text{[as fix-pt.]}\\
&= T\pi_2 \circ (\tau_{C,D}\langle id, h\rangle \oplus \mu f.\,(\eta \oplus f \diamond \tau_{C,D}\langle \pi_1, g\rangle) \diamond\\
&\quad \tau_{C,D}\langle \pi_1, g\rangle \diamond \tau_{C,D}\langle id, h\rangle) && \text{[by 1.2, 1.3]}\\
&= h \oplus T\pi_2 \circ \mu f.\,(\tau_{C,D}\langle \pi_1, g\rangle \diamond \tau_{C,D}\langle id, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) && \text{[by 3.3, 1.6]}\\
&= h \oplus T\pi_2 \circ \mu f.\,(\tau_{C,D}\langle \mathrm{id}, g \diamond \tau_{C,D}\langle id, h\rangle\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) && \text{[by 1.8]}
\end{aligned}$$

The latter term in this calculation is exactly $[\![\Gamma \rhd p + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow p; q)\,\mathrm{in}\, q^\star : TA]\!]$.

Axiom **(init)**. Let $h := [\![\Gamma \rhd p : TA]\!]$ and $g := [\![\Gamma, y : A \rhd q : TB]\!]$ but note that the metavariable r on the left and on the right of **(init)** corresponds to two different terms in context: the left r corresponds to $(\Gamma, x : B \rhd r : TB)$, whereas the right one corresponds to $(\Gamma, y : A, x : B \rhd r : TB)$. Let $e := [\![\Gamma, x : B \rhd r : TB]\!]$. Then due to the side condition $y \notin \mathrm{Vars}(x)$, $[\![\Gamma, y : A, x : B \rhd r : TB]\!] = e(\pi_1 \times \mathrm{id})$.

Let us show the following commutation property:

$$\eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,E}\langle \pi_1, e(\pi_1 \times \mathrm{id})\rangle = \tau_{C,E}\langle \pi_1, e\rangle \diamond \eta(\pi_1 \times \mathrm{id}). \tag{3.11}$$

By (1.5), (1.4) the left-hand side of (3.11) is equal to $T(\pi_1 \times \mathrm{id}) \circ \tau_{C\times D,E}\langle \pi_1, e(\pi_1 \times \mathrm{id})\rangle$ from which we obtain by naturality of strength $\tau_{C,E}\langle \pi_1\pi_1, e(\pi_1 \times \mathrm{id})\rangle$. On the other hand, by (1.7), the right-hand side of (3.11) is equal to $\tau_{C,E}\langle \pi_1, e\rangle \diamond \tau_{C,E}\langle \pi_1, \eta\rangle$, which by (1.8) is again equal to $\tau_{C,E}\langle \pi_1\pi_1, e(\pi_1 \times \mathrm{id})\rangle$. Now the proof of **(init)** runs as follows:

$$\begin{aligned}
&[\![\mathrm{do}\ y \leftarrow p; (\mathrm{init}\, x \leftarrow q \,\mathrm{in}\, r^\star)]\!]\\
&\quad = T\pi_2 \circ \mu f.\,(\tau_{C\times D,E}\langle \mathrm{id}, h\rangle \oplus\\
&\qquad \tau_{C\times D,E}\langle \pi_1, e(\pi_1 \times \mathrm{id})\rangle \diamond f) \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle && \text{[by def. of } [\![\_]\!]\text{]}\\
&\quad = \eta\pi_2 \diamond \eta(\pi_1 \times \mathrm{id}) \diamond \mu f.\,(\tau_{C\times D,E}\langle \mathrm{id}, h\rangle \oplus\\
&\qquad \tau_{C\times D,E}\langle \pi_1, e(\pi_1 \times \mathrm{id})\rangle \diamond f) \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle && \text{[by 1.5, 1.2]}\\
&\quad = \eta\pi_2 \diamond \mu f.\,(\eta(\pi_1 \times \mathrm{id}) \oplus f \diamond \tau_{C\times D,E}\langle \pi_1, e(\pi_1 \times \mathrm{id})\rangle) \diamond\\
&\qquad \tau_{C\times D,E}\langle \mathrm{id}, h\rangle \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle && \text{[by 3.1]}\\
&\quad = \eta\pi_2 \diamond \mu f.\,(\eta(\pi_1 \times \mathrm{id}) \oplus \tau_{C,E}\langle \pi_1, e\rangle \diamond f) \diamond && \text{[by 3.11,}\\
&\qquad \tau_{C\times D,E}\langle \mathrm{id}, h\rangle \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle && \text{Lemma 3.7]}\\
&\quad = \eta\pi_2 \diamond \mu f.\,(\eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,E}\langle \mathrm{id}, h\rangle \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle \oplus
\end{aligned}$$

$$\begin{aligned}
&\tau_{C,E}\langle \pi_1, e\rangle \diamond f) && \text{[by 3.3]}\\
&= \eta\pi_2 \diamond \mu f.\,(T(\pi_1 \times \mathrm{id}) \circ \tau_{C\times D,E}\langle \mathrm{id}, h\rangle \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle \oplus \\
&\quad \tau_{C,E}\langle \pi_1, e\rangle \diamond f) && \text{[by 1.5, 1.4]}\\
&= \eta\pi_2 \diamond \mu f.\,(\tau_{C,E}\langle \pi_1, h\rangle \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle \oplus \tau_{C,E}\langle \pi_1, e\rangle \diamond f) && \text{[by nat. of } \tau]\\
&= T\pi_2 \circ \mu f.\,(\tau_{C,E}\langle \mathrm{id}, h \diamond \tau_{C,D}\langle \mathrm{id}, g\rangle\rangle \oplus \tau_{C,E}\langle \pi_1, e\rangle \diamond f) && \text{[by 1.8]}\\
&= [\![\mathrm{init}\, x \leftarrow (\mathrm{do}\ y \leftarrow p; q)\, \mathrm{in}\, r^\star]\!] && \text{[by def. of } [\![\_]\!]]
\end{aligned}$$

Rule **(ind$_1$)**. Suppose that the premise is valid. We have thus $g \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \leqslant h$ where $h = [\![\Gamma \rhd p : TA]\!]$ and $g = [\![\Gamma, x : A \rhd q : TA]\!]$. By definition of the ordering $\leqslant$, this boils down to the identity $h = h \oplus g \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle$. Therefore

$$\begin{aligned}
\tau_{C,D}\langle \mathrm{id}, h\rangle &= \tau_{C,D}\langle \mathrm{id}, h \oplus g \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle\\
&= \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \mathrm{id}, g \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle\rangle && \text{[by 2.2]}\\
&= \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle && \text{[by 1.8]}
\end{aligned}$$

which means that $\tau_{C,D}\langle \mathrm{id}, h\rangle$ is a fixed point of $f \mapsto \tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f$. Hence $\tau_{C,D}\langle \mathrm{id}, h\rangle \geqslant \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)$. We obtain the claim by applying $T\pi_2$ to the both sides of this inequality.

Rule **(ind$_2$)**. Let $g := [\![\Gamma, x : A \rhd q : TA]\!]$, $h := [\![\Gamma \rhd p : TA]\!]$ and $e := [\![\Gamma, x : A \rhd r : TB]\!]$. Since r does not depend on y, $[\![\Gamma, y : A, x : A \rhd r : TB]\!] = e(\pi_1 \times \mathrm{id}) = e \diamond \eta(\pi_1 \times \mathrm{id})$. Then the premise results in the inequality $e \diamond \eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,D}\langle \mathrm{id}, g\rangle \leqslant e$. We conclude that e must be a fixed point of $f \mapsto e \oplus f \diamond \eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,D}\langle \mathrm{id}, g\rangle$ and hence $e \geqslant \mu f.\,(e \oplus f \diamond \eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,D}\langle \mathrm{id}, g\rangle)$. Therefore we have:

$$\begin{aligned}
&e \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle\\
&\geqslant \mu f.\,(e \oplus f \diamond \eta(\pi_1 \times \mathrm{id}) \diamond \tau_{C\times D,D}\langle \mathrm{id}, g\rangle) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle\\
&= \mu f.\,(e \oplus f \diamond T(\pi_1 \times \mathrm{id}) \circ \tau_{C\times D,D}\langle \mathrm{id}, g\rangle) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle && \text{[by 1.5, 1.4]}\\
&= \mu f.\,(e \oplus f \diamond \tau_{C,D}\langle \pi_1, g\rangle) \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle && \text{[by nat. of } \tau]\\
&= e \diamond \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f) && \text{[by 3.1]}\\
&= e \diamond \tau_{C,D}\langle \mathrm{id}, T\pi_2 \circ \mu f.\,(\tau_{C,D}\langle \mathrm{id}, h\rangle \oplus \tau_{C,D}\langle \pi_1, g\rangle \diamond f)\rangle && \text{[by Cor. 3.9]}
\end{aligned}$$

It can easily be seen that the established inequality is precisely the interpretation of the conclusion of **(ind$_2$)**, and thus we are done.

Completeness. We modify the monad construction from Theorem 2.7, specifically, we redefine $\sim$ to be the provable equivalence in ME* under the set of axioms Φ. Let us define the two necessary fixed points by the assignments:

$$\begin{aligned}
\mu f.\,(g \oplus h \diamond f) &:= [x : A \rhd \mathrm{init}\, y \leftarrow p\, \mathrm{in}\, q^\star : TB]_\sim,\\
\mu f.\,(e \oplus f \diamond h) &:= [x : B \rhd \mathrm{do}\ z \leftarrow (\mathrm{init}\, y \leftarrow \mathrm{ret}\, x\, \mathrm{in}\, q^\star); r : TC]_\sim.
\end{aligned}$$

where $g := [x : A \rhd p : TB]_\sim$, $h := [y : B \rhd q : TB]_\sim$ and $e := [z : B \rhd r : TC]_\sim$. Let us prove that these are indeed least fixed points. Consider the first one. We have:

$$\begin{aligned}
&\mu f.\,(g \oplus h \diamond f)\\
&\quad = [x : A \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim\\
&\quad = [x : A \rhd p + \mathsf{do}\ y \leftarrow (\mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star); q : TB]_\sim && \text{[by } \mathbf{(unf_1)}\text{]}\\
&\quad = [A : x \rhd p : TB]_\sim \oplus [t : TB \rhd \mathsf{do}\ y \leftarrow t; q : TB]_\sim \circ\\
&\quad\quad\ [A : x \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim && \text{[by def. of } \oplus, \circ\text{]}\\
&\quad = [A : x \rhd p : TB]_\sim \oplus [y : B \rhd q : TB]_\sim^\dagger \circ\\
&\quad\quad\ [A : x \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim && \text{[by def. of } \_^\dagger\text{]}\\
&\quad = g \oplus h \diamond \mu f.\,(g \oplus h \diamond f) && \text{[by 1.5]}
\end{aligned}$$

Therefore $\mu f.\,(g \oplus h \diamond f)$ is a fixed point of the map $f \mapsto g \oplus h \diamond f$. In order to prove that $\mu f.\,(g \oplus h \diamond f)$ is the least one, assume that some $[x : A \rhd s : TB]_\sim$ is another fixed point of the same map. By definition, $[x : A \rhd s : TB]_\sim = [x : A \rhd p + \mathsf{do}\ y \leftarrow s; q : TB]_\sim$ which results in the following inequalities:

$$[x : A \rhd s : TB]_\sim \geqslant [x : A \rhd p]_\sim, \tag{3.12}$$

$$[x : A \rhd s : TB]_\sim \geqslant [x : A \rhd \mathsf{do}\ y \leftarrow s; q : TB]_\sim. \tag{3.13}$$

By (3.12) and **(ind$_1$)**, $[x : A \rhd s : TB]_\sim \geqslant [x : A \rhd \mathsf{init}\, y \leftarrow s \,\mathsf{in}\, q^\star : TB]_\sim$ which by (3.13) is in turn greater than $[x : A \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim = \mu f.\,(g \oplus h \diamond f)$. Therefore, $\mu f.\,(g \oplus h \diamond f)$ is indeed the least fixed point. Analogously, by appealing to **(unf$_2$)** and **(ind$_2$)** one can prove that $\mu f.\,(e \oplus f \diamond h)$ makes the least fixed point of $f \mapsto e \oplus f \diamond h$.

The proof of the property (3.1) is as follows:

$$\begin{aligned}
&\mu f.\,(e \oplus f \diamond h) \diamond g\\
&\quad = [x : B \rhd \mathsf{do}\ z \leftarrow (\mathsf{init}\, y \leftarrow \mathsf{ret}\, x \,\mathsf{in}\, q^\star); r : TC]_\sim \diamond\\
&\quad\quad\ [x : A \rhd p : TB]_\sim\\
&\quad = [t : TB \rhd \mathsf{do}\ x \leftarrow t; z \leftarrow (\mathsf{init}\, y \leftarrow \mathsf{ret}\, x \,\mathsf{in}\, q^\star); r : TC]_\sim \circ\\
&\quad\quad\ [x : A \rhd p : TB]_\sim && \text{[by 1.5, def. of } \_^\dagger\text{]}\\
&\quad = [x : A \rhd \mathsf{do}\ x \leftarrow p; z \leftarrow (\mathsf{init}\, y \leftarrow \mathsf{ret}\, x \,\mathsf{in}\, q^\star); r : TC]_\sim && \text{[by def. of } \circ\text{]}\\
&\quad = [x : A \rhd \mathsf{do}\ z \leftarrow (\mathsf{do}\ x \leftarrow p; \mathsf{init}\, y \leftarrow \mathsf{ret}\, x \,\mathsf{in}\, q^\star); r : TC]_\sim && \text{[by } \mathbf{(assoc)}\text{]}\\
&\quad = [x : A \rhd \mathsf{do}\ z \leftarrow (\mathsf{init}\, y \leftarrow (\mathsf{do}\ x \leftarrow p; \mathsf{ret}\, x) \,\mathsf{in}\, q^\star); r : TC]_\sim && \text{[by } \mathbf{(init)}\text{]}\\
&\quad = [x : A \rhd \mathsf{do}\ z \leftarrow (\mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star); r : TC]_\sim && \text{[by } \mathbf{(unit_1)}\text{]}\\
&\quad = [t : TB \rhd \mathsf{do}\ z \leftarrow t; r : TC]_\sim \circ\\
&\quad\quad\ [x : A \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim && \text{[by def. of } \circ\text{]}\\
&\quad = [z : B \rhd r : TC]_\sim \diamond [x : A \rhd \mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star : TB]_\sim && \text{[by 1.5, def. of } \_^\dagger\text{]}
\end{aligned}$$

$$= e \diamond \mu f.\,(g \oplus h \diamond f)$$

Finally, let us prove the continuity of strength. We have by definition on one hand:

$$\begin{aligned}&\mu f.\,(\tau_{A,B}(\mathrm{id} \times g) \oplus \tau_{A,B}(\mathrm{id} \times h) \diamond f)\\&\quad = [x : A \rhd \mathrm{init}\langle x,y\rangle \leftarrow (\mathrm{do}\ y \leftarrow p; \mathrm{ret}\langle x,y\rangle)\ \mathrm{in}(\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star : T(A \times B)]_\sim\end{aligned}$$

and on the other hand:

$$\tau_{A,B}(\mathrm{id} \times \mu f.\,(g \oplus q \diamond h)) = [x : A \rhd \mathrm{do}\ y \leftarrow (\mathrm{init}\, y \leftarrow p\, \mathrm{in}\, q^\star); \mathrm{ret}\langle x,y\rangle : T(A \times B)]_\sim.$$

By Lemma 3.2 it suffices to establish the inequality:

$$\begin{aligned}&\mathrm{do}\ y \leftarrow (\mathrm{init}\, y \leftarrow p\, \mathrm{in}\, q^\star); \mathrm{ret}\langle x,y\rangle\\&\qquad\qquad \leqslant \mathrm{init}\langle x,y\rangle \leftarrow (\mathrm{do}\ y \leftarrow p; \mathrm{ret}\langle x,y\rangle)\ \mathrm{in}(\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star.\end{aligned} \tag{3.14}$$

Let us denote $(\mathrm{init}\langle x,y\rangle \leftarrow \mathrm{ret}\langle x,y\rangle\ \mathrm{in}(\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star)$ by s and prove that

$$\mathrm{do}\ y \leftarrow (\mathrm{init}\, y \leftarrow p\, \mathrm{in}\, q^\star); s \leqslant \mathrm{do}\ y \leftarrow p; s. \tag{3.15}$$

By **(ind$_2$)**, (3.15) amounts to the inequality

$$\mathrm{do}\ y \leftarrow (\mathrm{do}\ y \leftarrow p; q); s \leqslant \mathrm{do}\ y \leftarrow p; s$$

which is proven as follows:

$$\begin{aligned}
&\mathrm{do}\ y \leftarrow (\,\mathrm{do}\ y \leftarrow p; q); s\\
&\qquad = \mathrm{do}\ y \leftarrow (\mathrm{do}\ y \leftarrow p; q);\\
&\qquad\quad \mathrm{init}\langle x,y\rangle \leftarrow \mathrm{ret}\langle x,y\rangle\ \mathrm{in}(\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star && [\text{by def. of } s]\\
&\qquad = \mathrm{init}\langle x,y\rangle \leftarrow (\mathrm{do}\ y \leftarrow (\mathrm{do}\ y \leftarrow p; q); \mathrm{ret}\langle x,y\rangle)\ \mathrm{in}\\
&\qquad\quad (\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star && [\text{by } \textbf{(init)}]\\
&\qquad = \mathrm{init}\langle x,y\rangle \leftarrow (\mathrm{do}\ \langle x,y\rangle \leftarrow (\mathrm{do}\ y \leftarrow p; \mathrm{ret}\langle x,y\rangle); && [\text{by } \textbf{(assoc)},\\
&\qquad\quad y \leftarrow q; \mathrm{ret}\langle x,y\rangle)\ \mathrm{in}(\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star && \ \textbf{(unit}_2\textbf{)}]\\
&\qquad \leqslant \mathrm{init}\langle x,y\rangle \leftarrow (\mathrm{do}\ y \leftarrow p; \mathrm{ret}\langle x,y\rangle)\ \mathrm{in} && [\text{by } \textbf{(unf}_1\textbf{)},\\
&\qquad\quad (\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star && \ \text{def. of } \leqslant]\\
&\qquad = \mathrm{do}\ y \leftarrow p; \mathrm{init}\langle x,y\rangle \leftarrow \mathrm{ret}\langle x,y\rangle\ \mathrm{in}\\
&\qquad\quad (\mathrm{do}\ y \leftarrow q; \mathrm{ret}\langle x,y\rangle)^\star && [\text{by } \textbf{(init)}]\\
&\qquad = \mathrm{do}\ y \leftarrow p; s. && [\text{by def. of } s]
\end{aligned}$$

The inequality (3.15) that we just proved implies (3.14) as follows:

$$\begin{aligned}
&\mathsf{do}\ y \leftarrow (\,\mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star); \mathsf{ret}\langle x,y\rangle \\
&\quad\leqslant \mathsf{do}\ y \leftarrow (\mathsf{init}\, y \leftarrow p \,\mathsf{in}\, q^\star); && \text{[by } \textbf{(unf}_\mathbf{1}\textbf{)} \\
&\quad\ \mathsf{init}\langle x,y\rangle \leftarrow \mathsf{ret}\langle x,y\rangle \,\mathsf{in} (\mathsf{do}\ y \leftarrow q; \mathsf{ret}\langle x,y\rangle)^\star && \text{def. of} \leqslant\text{]} \\
&\quad\leqslant \mathsf{do}\ y \leftarrow p; \mathsf{init}\langle x,y\rangle \leftarrow \mathsf{ret}\langle x,y\rangle \,\mathsf{in} \\
&\quad\ (\mathsf{do}\ y \leftarrow q; \mathsf{ret}\langle x,y\rangle)^\star && \text{[by 3.15]} \\
&\quad\leqslant \mathsf{init}\langle x,y\rangle \leftarrow (\mathsf{do}\ y \leftarrow p; \mathsf{ret}\langle x,y\rangle) \,\mathsf{in} \\
&\quad\ (\mathsf{do}\ y \leftarrow q; \mathsf{ret}\langle x,y\rangle)^\star. && \text{[by } \textbf{(init)}\text{]}
\end{aligned}$$

The completeness part is thus proved. The proof of the theorem is completed. □

In addition to $\mathsf{ME}^\star$ we introduce a calculus ME^ω for ω-continuous monads. It is obtained from $\mathsf{ME}^\star$ by adding one more infinitary rule:

$$\textbf{(}\boldsymbol{\omega}\textbf{)}\ \frac{\forall i.\ \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i \,\mathsf{res}\, x \to r \leqslant t}{\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \,\mathsf{res}\, x \to r \leqslant t}$$

The premise should be read as the countable collection of inequalities with i ranging over non-negative integers.

Remark 3.13. It is easily seen that the induction rules **(ind$_1$)** and **(ind$_2$)** become derivable in ME^ω. E.g. let us show the first one. From $(\mathsf{do}\ x \leftarrow p; q) \leqslant p$ we can derive for arbitrary i, $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i) \leqslant p$ by applying **(unf$_2$)** sufficiently many times. Since $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i)$ is evidently provably equal to $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i \,\mathsf{res}\, x \to \mathsf{ret}\, x)$ by **(ω)**, we obtain $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \,\mathsf{res}\, x \to \mathsf{ret}\, x) \leqslant p$ or equivalently $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star) \leqslant p$. In the same way on can show **(ind$_2$)**.

Theorem 3.14. *The infinitary calculus ME^ω is sound and strongly complete over strong continuous Kleene monads.*

Proof. Soundness. By Theorem 3.12 we only need to establish soundness of the rule **(ω)**. Suppose the premise of **(ω)** is valid over some continuous strong Kleene monad $\mathbb{T}$. Let $C := [\![\Gamma]\!]$, $D := [\![A]\!]$, $E := [\![B]\!]$, $g := [\![\Gamma \rhd p : TA]\!]$, $h := [\![\Gamma, x : A \rhd q : TA]\!]$ and $e := [\![\Gamma, x : A \rhd r : TB]\!]$. We have:

$$\begin{aligned}
&[\![\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i \,\mathsf{res}\, x \to r]\!] \\
&\quad = [\![\mathsf{do}\ x \leftarrow (\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^i); r]\!] \\
&\quad = [\![\mathsf{do}\ x \leftarrow (\mathsf{do}\ x \leftarrow p; x \leftarrow q; \ldots ; q); r]\!] && \text{[by } \textbf{(assoc)}\text{]} \\
&\quad = e \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \ldots \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \tau_{C,E}\langle \mathrm{id}, g\rangle && \text{[by def. of } [\![\_]\!]\text{]}
\end{aligned}$$

which means that the premise of **(ω)** gives rise to the family of inequalities:

$$e \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \ldots \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \tau_{C,E}\langle \mathrm{id}, g\rangle \;\leqslant\; [\![\Gamma \rhd t : TB]\!]. \tag{3.16}$$

The conclusion of **(ω)** is now proved as follows:

$$\begin{aligned}
&[\![\mathrm{init}\, x \leftarrow p \,\mathrm{in}\, q^\star \,\mathrm{res}\, x \to r]\!] \\
&\quad = [\![\mathrm{do}\;\; x \leftarrow (\mathrm{init}\, x \leftarrow p \,\mathrm{in}\, q^\star); r]\!] \\
&\quad = e \diamond \tau_{C,D}\langle \mathrm{id}, T\pi_2 \circ \mu f.\, (\tau_{C,D}\langle id, g\rangle \oplus \tau_{C,D}\langle \pi_1, h\rangle \diamond f)\rangle && \text{[by def. of } [\![\_\,]\!]\text{]} \\
&\quad = e \diamond \mu f.\, (\tau_{C,D}\langle id, g\rangle \oplus \tau_{C,D}\langle \pi_1, h\rangle \diamond f) && \text{[by 3.9]} \\
&\quad = \;\sup\{e \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \ldots \diamond \tau_{C,D}\langle \mathrm{id}, h\rangle \diamond \tau_{C,E}\langle \mathrm{id}, g\rangle\} && \text{[by Def. 3.3]} \\
&\quad \leqslant [\![\Gamma \rhd t : TB]\!] && \text{[by 3.16]}
\end{aligned}$$

Completeness. We modify the construction from the completeness part of Theorem 3.12 by requiring the congruence $\sim$ to be stable under **(ω)**. The obtained model $\mathbb{K}_{\Sigma,\Phi}$ therefore makes a strong Kleene monad. We are left to prove ω-continuity.

By unfolding the fixed point in $e \diamond \mu f.\, (g \oplus h \diamond f)$ sufficiently many times we can show that $e \diamond \mu f.\, (g \oplus h \diamond f)$ is greater than any morphism of the form $e \diamond h \diamond \ldots \diamond h \diamond g$. Thus we are left to prove that whenever some u is greater than any $e \diamond h \diamond \ldots \diamond h \diamond g$ it is also greater than $e \diamond \mu f.\, (g \oplus h \diamond f)$. By construction of $\mathbb{K}_{\Sigma,\Phi}$, the morphisms g, h, e and u admit the following presentation: $g = [x : A \rhd p : TB]_\sim$, $h = [y : B \rhd q : TB]_\sim$, $e = [z : B \rhd r : TC]_\sim$, $u = [A : x \rhd t : TC]_\sim$. Then, by definition:

$$\begin{aligned}
e \diamond \mu f.\, (g \oplus h \diamond f) &= [x : A \rhd \mathrm{do}\;\; z \leftarrow (\mathrm{init}\, y \leftarrow p \,\mathrm{in}\, q^\star); r : TC]_\sim, \\
e \diamond h \diamond \ldots \diamond h \diamond g &= [x : A \rhd \mathrm{do}\;\; y \leftarrow p; y \leftarrow q; \ldots; y \leftarrow q; z \leftarrow q; r : TC]_\sim.
\end{aligned}$$

We have thus proved that $[x : A \rhd \mathrm{do}\;\; y \leftarrow p; y \leftarrow q; \ldots; y \leftarrow q; z \leftarrow q; r : TC]_\sim = [x : A \rhd \mathrm{do}\;\; z \leftarrow (\mathrm{init}\, y \leftarrow p \,\mathrm{in}\, q^i); r : TC]_\sim$ is smaller than $[A : x \rhd t : TC]_\sim$. Now the continuity follows from **(ω)**. □

The calculus ME^+ originally introduced for non-iterative programs, strictly speaking, cannot be used for proofs about programs involving Kleene star, because it lacks the corresponding congruence rule. Still, we agree to abuse notation slightly and use the entailment sign $\vdash_{\mathsf{ME}^+}$ to refer to provability in ME^+ endowed with the missing congruence rule **(cong_star)**. It can easily be seen that the rewriting rationale established for ME (Lemma 1.15) remains valid for MCE. Formally, it is a new fact, because we have extended the language with a new binding construction. Hence we restate it literally.

Lemma 3.15 (Rewriting rationale). *Let p, q be two programs with the same return type, and let Φ be a set of program equations. Then $\Phi \vdash_{\mathsf{ME}^+} p = q$ iff there is a sequence of programs $w_1, \ldots, w_n$ such that $p \doteq w_1$, $q \doteq w_n$ and for every $i < n$, w_i and w_{i+1} have forms*

$w_i \doteq C\{u\sigma\}$, $w_{i+1} \doteq C\{r\sigma\}$ for some context C, a variable substitution σ and programs u, r such that either $(u = r) \in \Phi$ or $(r = u) \in \Phi$ or $\vdash_{\mathsf{ME}^+} u = r$.

Proof. The proof is by minor adaptation of the proof of Lemma 1.15. □

3.3 Confluence of Kleene star unfolding

In the spirit of the rewriting approach taken in the previous sections we introduce a rewrite rule for unfolding a Kleene star:

$$\mathsf{k}\textbf{-rule:} \qquad \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star \;\rightarrowtail\; p + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow p; q) \,\mathsf{in}\, q^\star$$

The combined reduction relation $\rightarrowtail_{\mathsf{km}} := \rightarrowtail_{\mathsf{m}} \cup \rightarrowtail_{\mathsf{k}}$ evidently fails to terminate (because already the k-rule does not terminate), but it is still confluent. We prove this as soon as we have collected enough auxiliary facts about $\rightarrowtail_{\mathsf{k}}$ and $\rightarrowtail_{\mathsf{km}}$.

For every natural n we define an operator unf_n, distributing over all term constructors but the Kleene star, for which it is defined by the equation:

$$\begin{aligned} \mathsf{unf}_n(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star) \;=\; & \textstyle\sum_{i<n} \mathsf{init}\, x \leftarrow \mathsf{unf}_n(p) \,\mathsf{in}\, \mathsf{unf}_n(q)^i + \\ & \mathsf{init}\, x \leftarrow (\mathsf{init}\, x \leftarrow \mathsf{unf}_n(p) \,\mathsf{in}\, \mathsf{unf}_n(q)^n) \,\mathsf{in}\, \mathsf{unf}_n(q)^\star. \end{aligned}$$

Note that distributivity of unf_n over term constructors implies, in particular, that it is identical with respect to variables and constants.

Lemma 3.16. *For all programs p and q such that $p \rightarrowtail^\star_{\mathsf{k}} q$ and for every natural n greater than the length of the reduction sequence $p \rightarrowtail^\star_{\mathsf{k}} q$ there exists a reduction $q \rightarrowtail^\star_{\mathsf{k}} \mathsf{unf}_n(p)$.*

Proof. Let us introduce an auxiliary syntactical construction: $(\mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^{(\star+i)})$ and a counterpart of the k-rule for it:

$$\mathsf{k}'\textbf{-rule:} \qquad \mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^{(\star+i)} \;\rightarrowtail\; p + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow p; q) \,\mathsf{in}\, q^{(\star+i-1)}$$

where i is assumed to be greater than 0. The defined reduction relation is easily seen to be strongly normalising. A formal proof of this fact is as follows. Let h be a recursive function, assigning to every program a natural number in the following way: $h(op) := 1$, for every 0-ary term constructor, e.g. constant or variable, $h(\mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^{(\star+i)}) := h(s) + h(t) \cdot (i+1)$ and $h(op(t_1, \ldots, t_n)) := h(t_1) + \ldots + h(t_n)$ for the remaining term constructors. Let $\succ$ be the well-founded strict partial order over programs, defined by the equivalence $p \succ q \iff h(p) > h(q)$. By Lemma 2.9 the multiset extension of it $\succ_m$ is also well-founded. Let $\mathfrak{m}$ be the recursive operation used in Lemma 2.10, taking single programs to finite multisets of (+-free) programs. It is easy to see by definition that $p \rightarrowtail_{\mathsf{k}'} q$ implies $\mathfrak{m}(p) >_m \mathfrak{m}(q)$ which means that any infinite reduction $t_1 \rightarrowtail_{\mathsf{k}'} t_2 \rightarrowtail_{\mathsf{k}'} \ldots$ gives rise to an infinite chain $\mathfrak{m}(t_1) >_m \mathfrak{m}(t_2) >_m \ldots$. As we have seen, the latter is impossible. Therefore $\rightarrowtail_{\mathsf{k}'}$ is strongly normalising.

With this strong normalisation at hand, one easily concludes by Newman's Lemma (cf. e.g. [Klo92]) that the reduction relation $\rightarrowtail_{\mathsf{k}'}$ is confluent.

For every i let κ_i be the operator, recursively replacing every $(\mathsf{init}\, x \leftarrow t \,\mathsf{in}\, s^\star)$ with $(\mathsf{init}\, x \leftarrow t \,\mathsf{in}\, s^{(\star+i)})$ and let ι be an operator, recursively replacing any $(\mathsf{init}\, x \leftarrow t \,\mathsf{in}\, s^{(\star+i)})$ with $(\mathsf{init}\, x \leftarrow t \,\mathsf{in}\, s^\star)$. Evidently, for any i, $\iota \circ \kappa_i = \mathrm{id}$. Let $r = \kappa_n(p)$. Then the chain $p \rightarrowtail^\star_{\mathsf{k}} q$ can be step-wise simulated by a chain $r \rightarrowtail^\star_{\mathsf{k}'} u$ so that every intermediate element in $p \rightarrowtail^\star_{\mathsf{k}} q$ is the image under ι of the corresponding intermediate element of $r \rightarrowtail^\star_{\mathsf{k}'} u$. The reduction $r \rightarrowtail^\star_{\mathsf{k}'} u$ can be extended to $r \rightarrowtail^\star_{\mathsf{k}'} u \rightarrowtail^\star_{\mathsf{k}'} \mathsf{nf}_{\mathsf{k}'}(u) \equiv \mathsf{nf}_{\mathsf{k}'}(r)$. We will be done as soon as we prove that $\iota(\mathsf{nf}_{\mathsf{k}'}(r)) \equiv \mathsf{unf}_n(p)$. To that end we make use of the confluence of $\rightarrowtail_{\mathsf{k}'}$: w.l.o.g. we assume that $\mathsf{nf}_{\mathsf{k}'}(r)$ is obtained from r by innermost normalisation. This means that every redex $\left(\mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^{(\star+n)}\right)$ is replaced by the sum

$$\textstyle\sum_{i<n} \mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^i + \mathsf{init}\, x \leftarrow (\mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^n) \,\mathsf{in}\, t^\star$$

which is precisely $\mathsf{unf}_n(\mathsf{init}\, x \leftarrow s \,\mathsf{in}\, t^{(\star+n)})$. Innermost normalisation guarantees that the recursive definition of unf_n is simulated properly, and thus we are done. $\square$

If we take in the statement of the lemma $q := p$ and the reduction sequence $p \rightarrowtail^\star_{\mathsf{k}} q$ to be empty we immediately obtain

Corollary 3.17. *Every program p k-reduces to $\mathsf{unf}_n(p)$ for any n.*

In Chapter 1 we defined a reduction context as a term with at most one hole. More generally we call a *(reduction) multicontext* a term with many holes. We assume that the holes of a multicontext are ordered from left to right with respect to the string presentation of the corresponding term. Then, given a multicontext C with n holes and some terms $t_1, \ldots, t_n$, we denote by $C\{t_1, \ldots, t_n\}$ a term obtained by replacing the i-th hole with t_i. The notation $C\{t_1, \ldots, t_n\}$ shall always imply that **all** the holes of C are filled by the terms in brackets. More generally, $C\{C_1, \ldots, C_n\}$ denotes a multicontext whose holes are replaced by the multicontexts C_i accordingly. Let $\rightarrowtail$ be some reduction relation. We say that p *parallel reduces* to q under $\rightarrowtail$ and denote it $p \twoheadrightarrowtail q$ if for some multicontext C with at least one hole and some terms $t_1, \ldots, t_n, s_1, \ldots, s_n$, $p \doteq C\{t_1, \ldots t_n\}$, $q \doteq C\{s_1, \ldots s_n\}$ and for every i, $t_i \rightarrowtail s_i$ (cf. e.g. [BKdV03]). By definition, we have $\rightarrowtail \subseteq \twoheadrightarrowtail \subseteq \rightarrowtail^+$.

Lemma 3.18. *Let $p \twoheadrightarrowtail_{\mathsf{m}} s$ and $p \rightarrowtail_{\mathsf{k}} t$. Then there exists q such that $s \rightarrowtail^\star_{\mathsf{k}} q$ and $t \twoheadrightarrowtail_{\mathsf{m}} q$.*

Proof. Let $p \doteq C\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\}$ where C is some context and the term inside it is the k-redex. The k-reduction hence takes the form:

$$C\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\} \rightarrowtail C\{r + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow r; u)\,\mathsf{in}\, u^\star\}.$$

As the term $(\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star)$ cannot be a redex of any m-rule, there only two principal options for the placement of the m-redex.

The term $(\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star)$ *is a proper subterm of some* m*-redexes,* i.e. C can be presented in the form $K\{p_1, \ldots, p_{k-1}, M, p_{k+1}, \ldots, p_n\}$ where M is a context, $s \doteq K\{s_1, \ldots, s_n\}$ and for every i, $p_i \rightarrowtail_{\mathsf{m}} s_i$, with $p_k \doteq M\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\}$. W.l.o.g. we assume that p_k coincides with the m-redex, since any possible surrounding context can be turned into a part of K. It suffices to find such l that

$$M\{r + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow r; u)\,\mathsf{in}\, u^\star\} \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{m}} l \quad \text{and} \quad s_k \rightarrowtail^\star_{\mathsf{k}} l. \tag{3.17}$$

We consider only the situation when the reduction $p_k \rightarrowtail_{\mathsf{m}} s_k$ was performed according to the second unit law and omit the rather simpler verification of the remaining rules. The rewrite rule corresponding to the second unit law contains two metavariables. Hence it can only take one of the forms

$$\begin{aligned}&\mathsf{do}\;\; y \leftarrow \mathsf{ret}\, w; N\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\} \rightarrowtail_{\mathsf{m}} N[w/y]\{\mathsf{init}\, x \leftarrow r[w/y]\,\mathsf{in}\, u[w/y]^\star\},\\ &\mathsf{do}\;\; y \leftarrow \mathsf{ret}\, N\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\}; w \rightarrowtail_{\mathsf{m}} w[N\{\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star\}/y]\end{aligned}$$

where N is some reduction context. We put

$$l := N[w/y]\{r[w/y] + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow r[w/y]; u[w/y])\,\mathsf{in}\, u[w/y]^\star\}$$

in the first case and

$$l := w[N\{r + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow r; u)\,\mathsf{in}\, u^\star\}/y]$$

in the second case. The condition (3.17) is evidently satisfied.

The term $(\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star)$ *is not a subterm of any* m*-redex.* Then p can be presented in the form $K\{p_1, \ldots, p_{k-1}, (\mathsf{init}\, x \leftarrow r\,\mathsf{in}\, u^\star), p_{k+1}, \ldots, p_n\}$ where the terms p_i and the multi-context K are such that $s \doteq K\{s_1, \ldots, s_{k-1}, (\mathsf{init}\, x \leftarrow w\,\mathsf{in}\, v^\star), s_{k+1}, \ldots, s_n\}$, for every $i \neq k$, $p_i \rightarrowtail_{\mathsf{m}} s_i$, $r \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{m}} w$ and $u \rightarrowtail\!\!\!\!\twoheadrightarrow_{\mathsf{m}} v$. In this case we put

$$q := K\{s_1, \ldots, s_{k-1}, (w + \mathsf{init}\, x \leftarrow (\mathsf{do}\;\; x \leftarrow w; v)\,\mathsf{in}\, v^\star), s_{k+1}, \ldots, s_n\}$$

which completes the proof of the lemma. □

Corollary 3.19. *The reduction relations $\rightarrowtail_{\mathsf{m}}^{+}$ and $\rightarrowtail_{\mathsf{k}}^{\star}$ commute, i.e. for all programs p, s, t whenever $p \rightarrowtail_{\mathsf{m}}^{+} s$ and $p \rightarrowtail_{\mathsf{k}}^{\star} t$ there must exist q such that $s \rightarrowtail_{\mathsf{k}}^{\star} q$ and $t \rightarrowtail_{\mathsf{m}}^{\star} q$.*

Proof. Since $\rightarrowtail_{\mathsf{m}} \subseteq ⤕_{\mathsf{m}} \subseteq \rightarrowtail_{\mathsf{m}}^{+}$ we immediately conclude from the lemma that whenever $p \rightarrowtail_{\mathsf{m}} s$ and $p \rightarrowtail_{\mathsf{k}} t$, there exists q such that $s \rightarrowtail_{\mathsf{k}} q$ and $t \rightarrowtail_{\mathsf{m}}^{\star} q$. By straightforward induction over the length of the chain of k-reductions, we generalise this statement to:

whenever $p \rightarrowtail_{\mathsf{m}} s$ and $p \rightarrowtail_{\mathsf{k}}^{\star} t$, there exists q such that $s \rightarrowtail_{\mathsf{k}}^{\star} q$ and $t \rightarrowtail_{\mathsf{m}}^{+} q$.

This in turn can be generalised by induction over the length of the chain of m-reduction so as to obtain the claim. □

Lemma 3.20. *For some p and q, let $p \rightarrowtail_{\mathsf{k}}^{\star} q$. Then $\mathsf{nf}(q)$ can be reached from p by a reduction sequence of the form:*

$$p \doteq w_1 \rightarrowtail_{\mathsf{m}}^{\star} \mathsf{nf}(w_1) \rightarrowtail_{\mathsf{k}} w_2 \rightarrowtail_{\mathsf{m}}^{\star} \mathsf{nf}(w_2) \rightarrowtail_{\mathsf{k}} \ldots \rightarrowtail_{\mathsf{k}} w_n \doteq \mathsf{nf}(q). \tag{3.18}$$

Proof. Let us inductively form a sequence of reductions: $q_1 \rightarrowtail_{\mathsf{m}}^{\star} \ldots \rightarrowtail_{\mathsf{m}}^{\star} q_n$ in parallel with the programs $w_1, \ldots, w_n$ and such that for every k, $w_k \rightarrowtail_{\mathsf{k}}^{\star} q_k$. Let $q_1 := q$. Suppose we have constructed the w_k and q_k for every $k \leqslant i$. If $w_i \doteq q_i$ or $w_i \rightarrowtail_{\mathsf{k}} q_i$ then $n := i$ and we are finished. Consider the remaining case, i.e. for some w, $w_i \rightarrowtail_{\mathsf{k}} w \rightarrowtail_{\mathsf{k}}^{+} q_i$. If w_i is normal, then we put $w_{i+1} := w$ and $q_{i+1} := q_i$. If w_i is not normal, then by Corollary 3.19 there is u such that $\mathsf{nf}(w_i) \rightarrowtail_{\mathsf{k}}^{\star} u$ and $q_i \rightarrowtail_{\mathsf{m}}^{+} u$. In this case, let $q_{i+1} := u$ and let w_{i+1} be equal to the second program in the sequence $\mathsf{nf}(w_i) \rightarrowtail_{\mathsf{k}}^{\star} u$ if it contains at least one reduction and $q_{i+1} := u$ otherwise. By construction, (3.18) indeed holds for the w_k. The termination argument is based on the observation that, by construction, we cannot obtain an infinite sequence $q_i \doteq q_{i+1} \doteq \ldots$ and on strong normalisability of $\rightarrowtail_{\mathsf{m}}$. □

Theorem 3.21 (Confluence)**.** *The reduction relation $\rightarrowtail_{\mathsf{km}}$ is confluent.*

Proof. We obtain the claim by Hindley-Rosen Lemma [Hin64]. To that end we need to make sure that any pair of relations from $\{\rightarrowtail_{\mathsf{m}}, \rightarrowtail_{\mathsf{k}}\}^2$ commutes. By Corollary 3.19 this amounts to proving that both $\rightarrowtail_{\mathsf{m}}$ and $\rightarrowtail_{\mathsf{k}}$ individually are confluent. The confluence of the former relation has already been argued in Theorem 2.17 for the non-iterative case. The argument does not change in essence when extended to the MCE settings. In order to show the confluence of the latter relation, suppose for some p, s, t, $p \rightarrowtail_{\mathsf{k}}^{\star} s$ and $p \rightarrowtail_{\mathsf{k}}^{\star} t$. Let n be the maximal number of reduction steps in $p \rightarrowtail_{\mathsf{k}}^{\star} s$ and $p \rightarrowtail_{\mathsf{k}}^{\star} t$. By Lemma 3.16, $s \rightarrowtail_{\mathsf{k}}^{\star} \mathsf{unf}_n(p)$ and $s \rightarrowtail_{\mathsf{k}}^{\star} \mathsf{unf}_n(p)$. This establishes a confluence of $\rightarrowtail_{\mathsf{k}}$ and thus completes the proof of the theorem. □

3.4 Free strong (continuous) Kleene monads

In this section we require the underlying signature Σ to be plain (see Definition 1.31).

Theorem 2.18 provides a simple algorithm for deciding equality of non-iterative programs. Moreover it makes easy achieving metatheoretical results of the calculus for additive monads. E.g. we can easily show that the equality $(\mathrm{do}\ x \leftarrow a; p) = (\mathrm{do}\ x \leftarrow a; q)$ with atomic a is provable in ME^{+} iff $\vdash_{\mathsf{ME}^{+}} p = q$. Indeed, if either $\mathsf{nf}(p) \equiv \mathrm{ret}\, x$ or $\mathsf{nf}(p) \equiv \mathrm{ret}\, e_E, x : E$ then by Theorem 2.18, $\mathsf{nf}(\mathrm{do}\ x \leftarrow a; q) \equiv \mathsf{nf}(\mathrm{do}\ x \leftarrow a; p) \equiv \mathsf{nf}(a)$ which is only possible if $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. The case, obtained from the previous one by swapping p and q, follows by symmetry. For the remainder we have: $\mathsf{nf}(\mathrm{do}\ x \leftarrow a; p) \equiv (\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \mathsf{nf}(p))$, $\mathsf{nf}(\mathrm{do}\ x \leftarrow a; q) \equiv (\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \mathsf{nf}(q))$ and therefore, by Theorem 2.18, $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. In particular $p = q$ is provable in ME^{+}.

In the case of Kleene monads, we cannot use any similar approach, due to the infinitary character of the Kleene star. That is why proving very simple and intuitive statements like the following one turns out to be challenging.

Conjecture 3.1. Let for some programs p, q and $a \in \mathcal{A}_{\Sigma}$, $\vdash_{\mathsf{ME}^{\star}} \mathrm{do}\ x \leftarrow a; p = \mathrm{do}\ x \leftarrow a; q$. Then $\vdash_{\mathsf{ME}^{\star}} p = q$.

Our previous speculation, proving Conjecture 3.1 for non-iterative programs appeals to Theorem 2.18, characterising provable equality in ME^{+}. We can interpret this theorem as a result about the structure of the free strong additive monad. The fact that the unrestricted case of Conjecture 3.1 is rather more difficult means essentially that we do not have any appropriate characterisation of this kind for the free strong Kleene monad. In this section we establish some properties of free strong Kleene monads as well as of free strong continuous Kleene monads, whose case is rather simpler. These will imply Conjecture 3.1 for a broader class of programs than non-iterative ones, but the general case will remain open. We adhere the presentation in terms of provability in the calculi $\mathsf{ME}^{\star}$, ME^{ω}, implying the obvious parallel with the corresponding free structures.

The statement of the following lemma coincides with the statement of Lemma 1.30. Note, however, that we expanded the underlying language from ME to MCE.

Lemma 3.22. *Let t be some program.*

1. *If $h(t)$ is normal and $h \in \{\mathrm{fst}, \mathrm{snd}\}$ then both $h(t)$ and t are atomic.*

2. *If $f(t)$ is normal and $f \in \Sigma$ then both $f(t)$ and t are atomic.*

3. *If the return type of t is T-free and t is normal then t is atomic.*

Proof. The proof is obtained by some minor correction of the proof of Lemma 1.30 by including Kleene star. □

Remark 3.23. Like in the case of deterministic programs (cf. Remark 1.29), we can give a simple description of m-normal form. Specifically, every normal program p falls into the class P defined by the grammar:

$$\begin{aligned}P := \varnothing \mid A_\Sigma \mid \mathrm{ret}\,P \mid \langle P, P\rangle \mid P + P \mid \mathrm{init}\,x \leftarrow P\,\mathrm{in}\,P^\star \mid\\ \mathrm{do}\ \ x \leftarrow (\mathrm{init}\,y \leftarrow P\,\mathrm{in}\,P^\star); P \mid \mathrm{do}\ \ x \leftarrow A_\Sigma; P.\end{aligned}$$

Lemma 3.24. *Let a be a normal atomic program whose return type is computational. Then a cannot contain proper subterms with a computational return type.*

Proof. Let a be a normal atomic program with a computational return type. Suppose, on the contrary to the claim, there is a proper subterm b of a whose return type is computational. Let S be the smallest set of terms such that $b \in S$ and $\langle s, r\rangle \in S$ whenever $\langle s, r\rangle$ is a subterm of a and either $s \in S$ or $r \in S$. Let c be the maximal element of S with respect to the subterm relation. Note that c has a non-T-free return type. If $c \doteq a$ then for typing reasons c must be equal to b. Therefore, b coincides with a, and thus it is not a proper subterm of a.

Let c be distinct from a, i.e. c is a proper subterm of a. Then c should be an immediate subterm of some subterm d of a. Since Σ is supposed to be plain, d cannot be equal to $f(c)$ for any $f \in \Sigma$. By normality, d can be neither $\mathrm{fst}(c)$ nor $\mathrm{snd}(c)$. Since c is the maximal element of S, d cannot have forms $\langle d, r\rangle$ or $\langle r, d\rangle$ for any r. We have thus excluded all the possible options. Contradiction. □

Let us define an operator prg taking every program with a computational return type to a set of deterministic programs as follows:

- $\mathsf{prg}(p) := \{\mathsf{nf}(p)\}$ if $\mathsf{nf}(p)$ is atomic,
- $\mathsf{prg}(\varnothing) := \emptyset$,
- $\mathsf{prg}(p + q) := \mathsf{prg}(p) \cup \mathsf{prg}(q)$,
- $\mathsf{prg}(\mathrm{ret}\,p) := \{\mathrm{ret}\,p\}$,
- $\mathsf{prg}(\mathrm{do}\ \ x \leftarrow p; q) := \{\mathrm{do}\ \ x \leftarrow s; t \mid s \in \mathsf{prg}(p), t \in \mathsf{prg}(q)\}$,
- $\mathsf{prg}(\mathrm{init}\,x \leftarrow p\,\mathrm{in}\,q^\star) := \emptyset$.

Note that this definition is complete: for every $f \in \Sigma \cup \{\mathrm{fst}, \mathrm{snd}\}$ and every appropriate p, $\mathsf{nf}(f(p))$ is atomic by Lemma 3.24; any program of the form $\langle p, q\rangle$ is of non-computational type.

Essentially, prg flattens all nondeterminism on the top level, respecting the distributivity laws, and replaces the obtained nondeterministic sum by the set of its components. Now we can introduce the crucial definition of this section: for every program p, let

$$\gamma(p) := \mathsf{prg}(\mathsf{nf}(p)).$$

This function features a number of properties. The immediately apparent ones include: $\gamma(\varnothing) = \emptyset, \gamma(p+q) = \gamma(p) \cup \gamma(q), \gamma(\mathrm{init}\, x \leftarrow p \,\mathrm{in}\, q^\star) = \emptyset$. In order to state the further properties we need to introduce some auxiliary notions and notation.

Definition 3.25 (Shallow-deterministic programs)**.** We call a program *shallow-deterministic* if it is in either of the forms: $\mathrm{ret}\, s$, $(\mathrm{do}\ \ x \leftarrow a; t)$ or a, where a is normal atomic and t is shallow-deterministic.

It is easy to see that for every p with a computational return type, $\gamma(p)$ consists of shallow-deterministic programs. Note, however, that any program from $\gamma(p)$ is also normal, whereas in general shallow-deterministic programs must not be normal, e.g. $(\mathrm{do}\ \ x \leftarrow a; \mathrm{ret}\, x)$.

Let $\sim$ be an equivalence relation over programs. Then we can derive an equivalence relation $\hat{\sim}$ over shallow-deterministic programs based on $\sim$ by the clauses:

1. $a \mathrel{\hat{\sim}} a$ where $a \in \mathcal{A}_\Sigma$;
2. $(\mathrm{do}\ \ x \leftarrow a; \mathrm{ret}\, p) \mathrel{\hat{\sim}} a$ if $p \sim x$ where $a \in \mathcal{A}_\Sigma$;
3. $a \mathrel{\hat{\sim}} (\mathrm{do}\ \ x \leftarrow a; \mathrm{ret}\, q)$ if $q \sim x$ where $a \in \mathcal{A}_\Sigma$;
4. $(\mathrm{do}\ \ x \leftarrow a; p) \mathrel{\hat{\sim}} (\mathrm{do}\ \ x \leftarrow a; q)$ if $p \mathrel{\hat{\sim}} q$ where $a \in \mathcal{A}_\Sigma$;
5. $\mathrm{ret}\, p \mathrel{\hat{\sim}} \mathrm{ret}\, q$ if $p \sim q$.

It is straightforward to see that $\hat{\sim}$ is indeed an equivalence relation. Let us introduce some useful shorthands: $\simeq^\Phi_\star$ for $\hat{\sim}^\Phi_\star$ and $\simeq^\Phi_\omega$ for $\hat{\sim}^\Phi_\omega$ where $\sim^\Phi_\star$ and $\sim^\Phi_\omega$ are provable equivalence relations in $\mathsf{ME}^\star$ and ME^ω with the set of non-logical axioms Φ correspondingly. Moreover, we shortcut $\simeq^\emptyset_\star$ to $\simeq_\star$ and $\simeq^\emptyset_\omega$ to $\simeq_\omega$.

Let us denote by $\mathcal{S}_\Sigma$ the set of all sets of shallow-deterministic programs over Σ. We extend $\simeq^\Phi_\star$ and $\simeq^\Phi_\omega$ over $\mathcal{S}_\Sigma$ elementwise, e.g. we put $P \simeq^\Phi_\star Q$ if for every $p \in P$ there exists $q \in Q$ such that $p \simeq^\Phi_\star q$ and vice versa. We also define $\lesssim^\Phi_\star$ and $\lesssim^\Phi_\omega$ by the equivalences:

$$P \lesssim^\Phi_\star Q \iff P \cup Q \simeq^\Phi_\star Q \qquad \text{and} \qquad P \lesssim^\Phi_\omega Q \iff P \cup Q \simeq^\Phi_\omega Q.$$

Evidently, both the relations $\lesssim^\Phi_\star$ and $\lesssim^\Phi_\omega$ are preorders for which: $\lesssim^\Phi_\star \cap (\lesssim^\Phi_\star)^{-1} = \simeq^\Phi_\star$ and $\lesssim^\Phi_\omega \cap (\lesssim^\Phi_\omega)^{-1} = \simeq^\Phi_\omega$.

Since by definition γ always returns as the result a set of shallow-deterministic programs, the sets in the codomain of γ can be compared under $\simeq^{\Phi}_{\star}$, $\lesssim^{\Phi}_{\star}$, $\simeq^{\Phi}_{\omega}$ and $\lesssim^{\Phi}_{\omega}$.

Lemma 3.26. *Let p be a normal program distinct from $\varnothing$ and not of the form $u + r$ for any u, r. Then $\gamma(p) \neq \emptyset$ iff p is shallow-deterministic.*

Proof. If p is shallow-deterministic then, as directly follows from the definition of γ, $\gamma(p) = \{p\} \neq \emptyset$. Now assume that p is not shallow-deterministic. By assumption it is normal, hence p is of the form $(\mathrm{do}\ \bar{x} \leftarrow \bar{s}; t)$ where at least one term within $\{s_1, \dots, s_n, t\}$ is of the form $(\mathrm{init}\, y \leftarrow u\, \mathrm{in}\, r^{\star})$. By definition, we immediately have: $\gamma(p) = \mathsf{prg}(\mathsf{nf}(p)) = \mathsf{prg}(p) = \emptyset$. □

Lemma 3.27. *For some program p, let $\gamma(p) = \emptyset$. Then for any appropriately typed u and r, also $\gamma(\mathrm{do}\ x \leftarrow p; r) = \emptyset$ and $\gamma(\mathrm{do}\ x \leftarrow u; p) = \emptyset$.*

Proof. Let $\mathsf{nf}(p) \doteq \sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i$ be the presentation, described in Remark 3.23, i.e. all the p^i_j are atomic and all the q_i are normal. Then

$$\gamma(p) = \gamma(\mathsf{nf}(p)) = \textstyle\bigcup_i \gamma(\mathrm{do}\ \bar{x}^i \leftarrow p_i; q_i).$$

Since by assumption $\gamma(p) = \emptyset$, for every i, $\gamma(\mathrm{do}\ \bar{x}^i \leftarrow p_i; q_i) = \emptyset$. By Lemma 3.26 neither of the $(\mathrm{do}\ \bar{x}^i \leftarrow p_i; q_i)$ is shallow-deterministic, i.e. for every i, at least one of the p^i_j, q_i is of the form $(\mathrm{init}\, y \leftarrow s\, \mathrm{in}\, t^{\star})$.

By definition of γ, $\gamma(\mathrm{do}\ x \leftarrow p; r) = \gamma(w)$ where $w := \big(\sum_i \mathrm{do}\ \bar{x} \leftarrow \bar{p}^i; x \leftarrow q_i; r\big)$. More generally, w has form $\big(\sum_k \mathrm{do}\ \bar{z}^k \leftarrow \bar{s}^k; t_k\big)$ and the following condition is satisfied: for every k either $\mathsf{nf}(\mathrm{do}\ \bar{z} \leftarrow \bar{s}^k; t_k) \doteq \varnothing$ or at least one of the s^k_j, t_k has form $(\mathrm{init}\, y \leftarrow s\, \mathrm{in}\, t^{\star})$. It is easy to see that this condition is invariant under m-reductions. By definition, prg applied to any program satisfying this invariant returns $\emptyset$. In particular, $\mathsf{prg}(\mathsf{nf}(w)) = \gamma(w) = \emptyset$. We have thus proved that $\gamma(\mathrm{do}\ x \leftarrow p; r) = \emptyset$. The proof of the other identity runs analogously. □

Lemma 3.28. *Given two programs a and p where $a \in \mathcal{A}_{\Sigma}$,*

$$\gamma(\mathrm{do}\ x \leftarrow a; p) \simeq_{\star} \{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); t \mid t \in \gamma(p)\}.$$

Proof. Let $\mathsf{nf}(p) \doteq \sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i$ be the presentation, described in Remark 3.23. Then, by definition:

$$\begin{aligned}
&\gamma(\mathrm{do}\ x \leftarrow a; p)\\
&\quad = \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \mathsf{nf}(p))\\
&\quad = \gamma\Big(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \Big(\textstyle\sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i\Big)\Big)\\
&\quad = \textstyle\bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i).
\end{aligned}$$

We will be done once we prove that for every i:

$$\gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i) \simeq_\star \gamma\{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); t \mid t \in \gamma(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i)\}. \tag{3.19}$$

Indeed, then we would be able to prove the claim as follows:

$$\begin{aligned}
&\gamma(\mathrm{do}\ x \leftarrow a; p) \\
&\quad = \textstyle\bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i) \\
&\quad \simeq_\star \textstyle\bigcup_i \{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); t \mid t \in \gamma(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i)\} && \text{[by 3.19]} \\
&\quad \simeq_\star \Big\{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); t \mid t \in \gamma\Big(\textstyle\sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; q_i\Big)\Big\} \\
&\quad \simeq_\star \Big\{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); t \mid t \in \gamma(p)\Big\}.
\end{aligned}$$

Let us fix some i. By Lemma 3.26, if either of the p^i_j, q_i has form $(\mathrm{init}\, y \leftarrow s\, \mathrm{in}\, t^\star)$ then both sides (3.19) turn into $\emptyset$. Let us consider the remaining option. By definition, all the p^i_j must be atomic and q_i must be either atomic or of the form $\mathrm{ret}\, s$. If $|\bar{p}| > 0$ then one can conclude from the form of m-rules that $(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i)$ must be normal, and thus both sides of (3.19) evaluate to the singleton: $\{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i\}$. Finally, let $|\bar{p}^i| = 0$. If either $q_i \doteq x$ or $q_i \doteq \mathrm{ret}\, e_E$ where x has type $E \in \mathcal{U}$ then the left-hand side of (3.19) is equal to $\{\mathsf{nf}(a)\}$, the right-hand side of (3.19) is equal to $\{\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \mathrm{ret}\, x\}$, and these are equivalent under $\simeq_\star$ by definition. If $|\bar{p}^i| = 0$ but none of the cases we have considered applies, then $(\mathrm{do}\ x \leftarrow \mathsf{nf}(a); \bar{x}^i \leftarrow \bar{p}^i; q_i)$ must be normal and the proof of (3.19) runs identically to the case $|p^i| > 0$. $\square$

Lemma 3.29. *For all programs p and q, if $p \rightarrowtail^\star_{\mathsf{km}} q$ then $\gamma(p) \lesssim_\star \gamma(q)$.*

Proof. It suffices to prove the claim for the one-step reductions. The general case will then follow by induction over the length of the reduction chain. If $p \rightarrowtail_{\mathsf{m}} q$ then, by definition $\gamma(p) = \mathsf{prg}(\mathsf{nf}(p)) = \mathsf{prg}(\mathsf{nf}(q)) = \gamma(q)$ and we are done.

Consider the other case, i.e. $p \rightarrowtail_{\mathsf{k}} q$. In particular, we have $p \rightarrowtail_{\mathsf{km}} \mathsf{nf}(q)$. Since, as we have seen previously, $\gamma(\mathsf{nf}(q)) = \gamma(q)$, we will be done once we show the inequality: $\gamma(p) \lesssim_\star \gamma(\mathsf{nf}(q))$. By Lemma 3.20 and transitivity of $\lesssim_\star$ instead we can prove that $\gamma(p) \lesssim_\star \gamma(\mathsf{nf}(q))$, whenever $p \rightarrowtail^\star_{\mathsf{m}} \mathsf{nf}(p) \rightarrowtail_{\mathsf{k}} q$.

By Remark 3.23, for some programs s, t and some context C, $\mathsf{nf}(p) \doteq C\{\mathrm{init}\, x \leftarrow s\, \mathrm{in}\, t^\star\}$ and $q \doteq C\{s + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow s; t)\, \mathrm{in}\, t^\star\}$. As C must be normal, it can be presented in the form

$$C \equiv \textstyle\sum_i \mathrm{do}\ \bar{x} \leftarrow \bar{a}^i; b_i + \sum_j \mathrm{do}\ \bar{y} \leftarrow \bar{c}^j; \mathrm{ret}\, C_j + \sum_k \mathrm{do}\ \bar{z} \leftarrow \bar{K}^k; M_k + E$$

where all the a^i_l, c^j_l, b_i are atomic, all the K^k_l, M_k, C_j are contexts (no more than one of them indeed contains the hole), E is either $\square$ or $\varnothing$ and for every k at least one element of the sequence $(\bar{K}^k, M^k)$ has form $(\mathrm{init}\, y \leftarrow K \,\mathrm{in}\, M^\star)$. By the definition of γ for every k:

$$\begin{aligned}\gamma((\mathrm{do}\ \bar{z} \leftarrow \bar{K}^k; M_k)\{\mathrm{init}\, x \leftarrow s \,\mathrm{in}\, t^\star\}) &= \emptyset,\\ \gamma((\mathrm{do}\ \bar{z} \leftarrow \bar{K}^k; M_k)\{s + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow s; t) \,\mathrm{in}\, t^\star\}) &= \emptyset\,.\end{aligned}$$

Therefore, by the definitions of γ and $\simeq_\star$:

$$\begin{aligned}\gamma(p) &= \gamma(\mathsf{nf}(p))\\ &= \gamma\Big(\textstyle\sum_i \mathrm{do}\ \bar{x} \leftarrow \bar{a}^i; b_i\Big) \cup\\ &\quad\ \gamma\Big(\textstyle\sum_j \mathrm{do}\ \bar{x} \leftarrow \bar{c}^j; \mathrm{ret}\, C_j\{\mathrm{init}\, x \leftarrow s \,\mathrm{in}\, t^\star\}\Big) \cup \gamma\Big(E\{\mathrm{init}\, x \leftarrow s \,\mathrm{in}\, t^\star\}\Big)\\ &\simeq_\star \gamma\Big(\textstyle\sum_i \mathrm{do}\ \bar{x} \leftarrow \bar{a}^i; b_i\Big) \cup\\ &\quad\ \gamma\Big(\textstyle\sum_j \mathrm{do}\ \bar{x} \leftarrow \bar{c}^j; \mathrm{ret}\, C_j\{s + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow s; t) \,\mathrm{in}\, t^\star\}\Big) \cup\\ &\quad\ \gamma\Big(E\{s + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow s; t) \,\mathrm{in}\, t^\star\}\Big)\\ &\gtrsim_\star \gamma(q)\end{aligned}$$

and thus the proof is completed. □

Lemma 3.30. *Let for some p and q, $p \rightarrowtail_{\mathsf{m}} q$. Then for every natural n, $p \rightarrowtail_{\mathsf{k}}^\star . \rightarrowtail_{\mathsf{m}}^\star \mathsf{unf}_n(q)$.*

Proof. Let $p \doteq C\{s\}$ and $q \doteq C\{t\}$ where $s \rightarrowtail t$ is an m-rule instance and C some context. We proceed by induction over the complexity of C.

Induction Base: $C \doteq \square$. In this case $p \rightarrowtail_{\mathsf{m}} q$ is an instance of an m-reduction rule. In order to prove the claim we need to consider all these rules one by one. We restrict ourselves as usual only to the second unit law rule, as the remainder can be verified in a similar vein (but rather more easily). Suppose $p \doteq (\mathrm{do}\ x \leftarrow \mathrm{ret}\, s; t)$ and $q \doteq t[s/x]$. It easily follows from the definition of unf_n that $\mathsf{unf}_n(t[s/x]) \doteq \mathsf{unf}_n(t)[\mathsf{unf}_n(s)/x]$. The reduction in question is now as follows:

$$p \doteq (\mathrm{do}\ x \leftarrow \mathrm{ret}\, s; t) \rightarrowtail_{\mathsf{k}}^\star (\mathrm{do}\ x \leftarrow \mathrm{ret}\, \mathsf{unf}_n(s); \mathsf{unf}_n(t)) \rightarrowtail_{\mathsf{m}} \mathsf{unf}_n(t)[\mathsf{unf}_n(s)/x].$$

Induction Step. Unless C has a Kleene star on top we can easily switch to the induction hypothesis by distributivity of unf_n over the term constructors distinct from the Kleene star. For instance, if $C \doteq (\mathrm{do}\ x \leftarrow K; M)$ then the reduction sequence in question is:

$$\begin{aligned}p \doteq \mathrm{do}\ x \leftarrow K\{t\}; M\{t\} &\rightarrowtail_{\mathsf{k}}^\star . \rightarrowtail_{\mathsf{m}}^\star\\ &\qquad \mathrm{do}\ x \leftarrow \mathsf{unf}_n(K\{s\}); \mathsf{unf}_n(M\{s\}) \doteq \mathsf{unf}_n(\mathrm{do}\ x \leftarrow K\{s\}; M\{s\}).\end{aligned}$$

Let $C \doteq (\mathrm{init}\, x \leftarrow K \,\mathrm{in}\, M^\star)$. In this case we have

$$
\begin{aligned}
p \doteq \mathsf{init}\, x \leftarrow K\{t\}\, \mathsf{in}\, M\{t\}^\star \rightarrowtail_{\mathsf{k}}^\star &\\
\mathsf{init}\, x \leftarrow \mathsf{unf}_n(K\{t\})\, \mathsf{in}\, \mathsf{unf}_n(M\{t\})^n \rightarrowtail_{\mathsf{k}}^\star . \rightarrowtail_{\mathsf{m}}^\star &\\
\mathsf{do}\ x \leftarrow \mathsf{unf}_n(K\{s\}); \mathsf{unf}_n(M\{s\}) \doteq \mathsf{unf}_n(\mathsf{do}\ x \leftarrow K\{s\}; M\{s\}).&
\end{aligned}
$$

This completes the proof of the induction step, and thus the lemma is proved. □

Now let us introduce one more function, sending programs to $\mathcal{S}_\Sigma$. For every p we put by definition:

$$\mathsf{NF}(p) \ := \ \bigcup \{\gamma(q) \mid p \rightarrowtail_{\mathsf{km}}^\star q\}.$$

The intuition behind this definition is as follows. Given a program p, we rewrite it nondeterministically by applying m-rules and unfolding the Kleene star. Then at some point we ensure that the m-normalisation step was the latest and flatten the top-level nondeterminism under prg. By definition, $\mathsf{NF}(p)$ is the union of all sets of programs obtained in this way.

The construction of $\mathsf{NF}(p)$ brings a hope that every m-normal $t \leqslant p$ such that the nondeterministic operators (deadlock, choice, Kleene star) occur in t under ret's, is provably equivalent to some $s \in \mathsf{NF}(p)$.

Example 3.1. Let p be a program with a computational return type. If p is non-iterative, then $\mathsf{nf}(p) = \sum_{i \in I} p_i$ (possibly with $I = \emptyset$) where all p_i are shallow-deterministic (and moreover non-iterative). Then it is easy to see that $\mathsf{NF}(p) = \{p_i \mid i \in I\}$.

If p is shallow-deterministic, then by definition $\mathsf{NF}(p) = \{\mathsf{nf}(p)\}$.

Let p be of the form $(\mathsf{init}\, x \leftarrow q\, \mathsf{in}\, r^\star)$ where q and r do not contain 'ret'. Then it is easy to see that $\mathsf{NF}(p) = \{\mathsf{nf}(\mathsf{init}\, x \leftarrow s\, \mathsf{in}\, t^i) \mid s \in \mathsf{NF}(q), t \in \mathsf{NF}(r)\}$.

Our goal now is to show that the class of reduction chains $p \rightarrowtail_{\mathsf{km}}^\star q$ in the definition of $\mathsf{NF}(p)$ can be specified more precisely if we are allow identification modulo $\simeq_\star$. For example, we are going to see that we can replace the class of all reductions $p \rightarrowtail_{\mathsf{km}}^\star q$ by the reductions of the form $p \rightarrowtail_{\mathsf{k}}^\star \mathsf{unf}_n(p)$ (Lemma 3.31).

Lemma 3.31. *For any program p:*

$$\mathsf{NF}(p) \ \simeq_\star \mathsf{NF}(\mathsf{nf}(p)), \tag{3.20}$$

$$\mathsf{NF}(p) \ \simeq_\star \bigcup_i \gamma(\mathsf{unf}_i(p)). \tag{3.21}$$

Proof. For some q, let $p \rightarrowtail_{\mathsf{km}}^\star q$. Since for every p, $p \rightarrowtail_{\mathsf{m}}^\star \mathsf{nf}(p)$, by Theorem 3.21 there is r such that $\mathsf{nf}(p) \rightarrowtail_{\mathsf{km}}^\star r$ and $q \rightarrowtail_{\mathsf{km}}^\star r$. By Lemma 3.29, $\gamma(p) \lesssim_\star \gamma(q) \lesssim_\star \gamma(r)$. Now we have: $\mathsf{NF}(p) = \bigcup\{\gamma(q) \mid p \rightarrowtail_{\mathsf{km}}^\star q\} \lesssim_\star \bigcup\{\gamma(r) \mid \mathsf{nf}(p) \rightarrowtail_{\mathsf{km}}^\star r\} = \mathsf{NF}(\mathsf{nf}(p))$. On the other hand $\mathsf{NF}(\mathsf{nf}(p)) \lesssim_\star \bigcup\{\gamma(r) \mid p \rightarrowtail_{\mathsf{km}}^\star \mathsf{nf}(p) \rightarrowtail_{\mathsf{km}}^\star r\} \subseteq \mathsf{NF}(p)$ and thus we have (3.20) proved.

We prove the identity (3.21) by establishing mutual inequality. Note that the right-hand side of (3.21) is obtained from the union $\bigcup\{\gamma(q) \mid p \rightarrowtail^{\star}_{\mathsf{km}} q\}$ by cancelling the reductions not having form $p \rightarrowtail^{\star}_{\mathsf{k}} \mathsf{unf}_i(p)$. Therefore:

$$\bigcup_i \gamma(\mathsf{unf}_i(p)) \lesssim_\star \mathsf{NF}(p) = \bigcup\{\gamma(q) \mid p \rightarrowtail^{\star}_{\mathsf{km}} q\}.$$

We are left to prove that $\mathsf{NF}(p) \lesssim_\star \bigcup_i \gamma(\mathsf{unf}_i(p))$. By definition, it suffices to show that for every q such that $p \rightarrowtail^{\star}_{\mathsf{km}} q$ there exists i such that $\gamma(q) \lesssim_\star \gamma(\mathsf{unf}_i(p))$. We prove it by induction over n, the number of one-step m-reductions in $p \rightarrowtail^{\star}_{\mathsf{km}} q$. If $n = 0$ then we take as i the length of the sequence $p \rightarrowtail^{\star}_{\mathsf{km}} q$. By Lemma 3.16 $q \rightarrowtail^{\star}_{\mathsf{k}} \mathsf{unf}_i(p)$ and therefore, by Lemma 3.29 $\gamma(q) \lesssim_\star \gamma(\mathsf{unf}_i(p))$.

Suppose $n > 0$ and consider the last step of the reduction chain $p \rightarrowtail^{\star}_{\mathsf{km}} q$. If it corresponds to an m-reduction then there must exist r such that $p \rightarrowtail^{\star}_{\mathsf{km}} r \rightarrowtail_{\mathsf{m}} q$ and the number of one-step m-reductions in the sequence $p \rightarrowtail^{\star}_{\mathsf{km}} r$ is smaller than originally. By induction hypothesis, $\gamma(r) \lesssim_\star \gamma(\mathsf{unf}_i(p))$. By definition of γ, $\gamma(r) = \gamma(q)$, which proves the induction step. Suppose the last step in the chain $p \rightarrowtail^{\star}_{\mathsf{km}} q$ is by the k-rules. Let $u \rightarrowtail^{\star}_{\mathsf{k}} q$ be the longest tail of k-reductions in $p \rightarrowtail^{\star}_{\mathsf{km}} q$. Since $n > 0$ the original chain decomposes to

$$p \rightarrowtail^{\star}_{\mathsf{km}} r \rightarrowtail_{\mathsf{m}} u \rightarrowtail^{\star}_{\mathsf{k}} q. \tag{3.22}$$

By Lemma 3.16, assume w.l.o.g. that $q \doteq \mathsf{unf}_k(u)$, for some k. Then, by Lemma 3.30, there exists w such that (3.22) is equivalent to

$$p \rightarrowtail^{\star}_{\mathsf{km}} r \rightarrowtail^{\star}_{\mathsf{k}} w \rightarrowtail^{\star}_{\mathsf{m}} q.$$

By the induction hypothesis $\gamma(w) \lesssim_\star \gamma(\mathsf{unf}_i(p))$. Again, by definition of γ, $\gamma(w) = \gamma(q)$, which implies: $\gamma(q) \lesssim_\star \gamma(\mathsf{unf}_i(p))$ and thus the proof is complete. □

Lemma 3.32. *For all appropriately typed programs p, q, r and u*

$$\begin{aligned}&\mathsf{NF}(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q\, \mathrm{in}\, r^\star); u)\\ &\qquad\qquad \simeq_\star \bigcup_i \mathsf{NF}(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q\, \mathrm{in}\, r^i); u).\end{aligned} \tag{3.23}$$

Proof. First transform the left-hand side of (3.23) as follows.

$$\begin{aligned}&\mathsf{NF}(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q\, \mathrm{in}\, r^\star); u)\\ &\quad\simeq_\star \bigcup_i \gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q\, \mathrm{in}\, r^\star); u)) && \text{[by 3.21]}\\ &\quad\simeq_\star \bigcup_i \bigcup_{j<i} \gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q\, \mathrm{in}\, r^j); u)) \cup\\ &\qquad\quad \bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{unf}_i(p); y \leftarrow (\mathrm{init}\, z \leftarrow (\mathrm{init}\, z \leftarrow \mathsf{unf}_i(q)\, \mathrm{in}\\ &\qquad\qquad\quad \mathsf{unf}_i(r)^i)\, \mathrm{in}\, \mathsf{unf}_i(r)^\star); \mathsf{unf}_i(u)) && \text{[by def. of } \mathsf{unf}_i\text{]}\end{aligned}$$

It follows from Lemmas 3.26, 3.27 that the latter indexed union evaluates to the empty set. Therefore the goal simplifies down to

$$\begin{aligned}&\mathsf{NF}(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^\star); u)\\ &\qquad \simeq_\star \textstyle\bigcup_i \bigcup_{j<i} \gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u)).\end{aligned} \tag{3.24}$$

Let us show that the right-hand side of (3.24) is greater than

$$\gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u))$$

for any i, j (note, this is trivial when $j < i$). Let n be the length of the reduction sequence

$$\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u \rightarrowtail_{\mathsf{k}}^\star \mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u)$$

Then, by Lemma 3.16 for $k = \max\{j, n\} + 1$

$$\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u) \rightarrowtail_{\mathsf{k}}^\star \mathsf{unf}_k(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u)$$

Therefore

$$\begin{aligned}&\gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u))\\ &\quad \lesssim_\star \gamma(\mathsf{unf}_k(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u)) && \text{[by Lemma 3.29]}\\ &\quad \lesssim_\star \textstyle\bigcup_i \bigcup_{j<i} \gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; y \leftarrow (\mathrm{init}\, z \leftarrow q \,\mathrm{in}\, r^j); u))\end{aligned}$$

We have thus shown that we can drop the condition $j < i$ for the indexed union in (3.24), after which we are done by Lemma 3.21. □

Lemma 3.33. *Let for some programs q and r, $\vdash_{\mathsf{ME}^\omega} q = r$. Then for every $p_1, \ldots, p_n$, $\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, q) \simeq_\omega \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, r)$. If, moreover, $\vdash_{\mathsf{ME}^\star} q = r$ then $\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, q) \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, r)$.*

Proof. First, let us show that w.l.o.g. $n = 1$. By assumption, $\vdash_{\mathsf{ME}^\omega} q = r$ and therefore

$$\vdash_{\mathsf{ME}^\omega} q[\mathrm{pr}_1^n(z)/x_1, \ldots, \mathrm{pr}_n^n(z)/x_n] = r[\mathrm{pr}_1^n(z)/x_1, \ldots, \mathrm{pr}_n^n(z)/x_n] \tag{3.25}$$

If we had the claim proved with $n = 1$ we would prove the general case as follows:

$$\begin{aligned}&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, q)\\ &\quad \simeq_\omega \mathsf{NF}(\mathrm{do}\ z \leftarrow (\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, \bar{x}); \mathrm{ret}\, q[\mathrm{pr}_1^n(z)/x_1, \ldots, \mathrm{pr}_n^n(z)/x_n]) && \text{[by 3.20]}\\ &\quad \simeq_\omega \mathsf{NF}(\mathrm{do}\ z \leftarrow (\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, \bar{x}); \mathrm{ret}\, r[\mathrm{pr}_1^n(z)/x_1, \ldots, \mathrm{pr}_n^n(z)/x_n]) && \text{[by 3.25]}\\ &\quad \simeq_\omega \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{p}; \mathrm{ret}\, r) && \text{[by 3.20]}\end{aligned}$$

The same argument applies to the case $\vdash_{\mathsf{ME}^\star} q = r$.

We are left to prove that $\vdash_{\mathsf{ME}^\star} q = r \implies \mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, q) \simeq_\star \mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r)$ and $\vdash_{\mathsf{ME}^\omega} q = r \implies \mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, q) \simeq_\omega \mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r)$. We consider only the former implication because the latter one can be proved analogously. We have:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, q)\\
&\quad\simeq_\star \textstyle\bigcup_i \gamma(\mathsf{unf}_i(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, q)) && \text{[by 3.21]}\\
&\quad\simeq_\star \textstyle\bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{unf}_i(p); \mathrm{ret}\, \mathsf{unf}_i(q)) && \text{[by def. of } \mathsf{unf}_i\text{]}\\
&\quad\simeq_\star \textstyle\bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(q)) && \text{[by def. of } \gamma\text{].}
\end{aligned}$$

In the same way, $\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r) \simeq_\star \bigcup_i \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(r))$. Therefore it suffices to prove the equivalence

$$\gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(q)) \simeq_\star \gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(r))$$

for every i. Let us fix i. By m-normality, $\mathsf{nf}(\mathsf{unf}_i(p))$ has form $(\sum_j \mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; t_j)$. Since by (3.20), $\gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(q)) \simeq_\star \bigcup_j \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(q))$ and $\gamma(\mathrm{do}\ x \leftarrow \mathsf{nf}(\mathsf{unf}_i(p)); \mathrm{ret}\, \mathsf{unf}_i(r)) \simeq_\star \bigcup_j \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(r))$ we will be done as soon as we establish the equivalence:

$$\gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(q)) \simeq_\star \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(r))$$

for every t_j. By Lemma 3.27 both sides of this equivalence evaluate to $\emptyset$ if one of the $s^j_1, \ldots, s^j_{n_j}, t_j$ has a Kleene star on top. Consider the remaining case. By construction, all the s^j_k must be atomic. If t_j is also atomic then we are done by the definition of $\simeq_\star$ and the calculation: $\vdash_{\mathsf{ME}^\star} \mathsf{unf}_i(q) = q = r = \mathsf{unf}_i(r)$. Finally, consider the only remaining option: $t_j \doteq \mathrm{ret}\, w$ with some w. Then

$$\begin{aligned}
&\gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(q))\\
&\quad= \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow \mathrm{ret}\, w; \mathrm{ret}\, \mathsf{unf}_i(q))\\
&\quad\simeq_\star \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, w; \mathsf{unf}_i(q))) && \text{[by 3.20]}\\
&\quad\simeq_\star \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, w; \mathsf{unf}_i(r))) && \text{[by def. of } \simeq_\star\text{]}\\
&\quad\simeq_\star \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow \mathrm{ret}\, w; \mathrm{ret}\, \mathsf{unf}_i(q)) && \text{[by 3.20]}\\
&\quad= \gamma(\mathrm{do}\ \bar{x}^j \leftarrow \bar{s}^j; x \leftarrow t_j; \mathrm{ret}\, \mathsf{unf}_i(q)).
\end{aligned}$$

In this calculation we made use of the following fact:

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ x \leftarrow \mathrm{ret}\, w; \mathsf{unf}_i(q) = \mathrm{do}\ x \leftarrow \mathrm{ret}\, w; \mathsf{unf}_i(r)$$

whose proof is as follows:

$$\begin{aligned}
&\mathrm{do}\ x \leftarrow \mathrm{ret}\, w; \mathsf{unf}_i(q)\\
&\quad= \mathrm{do}\ x \leftarrow \mathrm{ret}\, w; q && \text{[by def. of } \mathsf{unf}_i\text{]}
\end{aligned}$$

$$
\begin{aligned}
&= \mathrm{do}\ x \leftarrow \mathsf{ret}\, w; r && \text{[by assm.]}\\
&= \mathrm{do}\ x \leftarrow \mathsf{ret}\, w; \mathsf{unf}_i(r). && \text{[by def. of } \mathsf{unf}_i\text{]}
\end{aligned}
$$

The proof is now completted. □

Lemma 3.34. *Let Φ be a set of program equations and let $\simeq$ be an equivalence relation over $\mathcal{S}_\Sigma$ such that $\simeq$ is weaker than $\simeq^\Phi_\star$ and for every $(s = t) \in \Phi$, all programs $u_1, \ldots, u_n, r$ and every normal $a \in \mathcal{A}_\Sigma$:*

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[s/v]; r) \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[t/v]; r). \tag{3.26}$$

Let p and q be two programs such that $\Phi \vdash_{\mathsf{ME}^\star} p = q$. Then for every $u_1, \ldots, u_n, r$ and for every normal $a \in \mathcal{A}_\Sigma$:

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r). \tag{3.27}$$

Proof. Note for the future that since $\simeq$ is weaker than $\simeq_\star$, it suffices to prove

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r). \tag{3.28}$$

instead of (3.27). We will be done once we prove the claim in two cases: if $p = q$ is a logical axiom of $\mathsf{ME}^\star$, and if $p = q$ is obtained from Φ in one step. The general case would then follow by induction over the proof complexity of $\Phi \vdash_{\mathsf{ME}^\star} p = q$.

For the axioms of ME^+ (3.28) and thus (3.27) follows from (3.20). Indeed, if $p = q$ is an axiom of ME^+ then $\vdash_{\mathsf{ME}^+} a[p/v] = a[q/v]$. Therefore, by Theorem 2.18, $\mathsf{nf}(a[p/v]) \equiv \mathsf{nf}(a[q/v])$. Then we have by (3.20) and the definition of nf:

$$
\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r)\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[p/v]); r)\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[q/v]); r)\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r).
\end{aligned}
$$

The remaining three axioms of $\mathsf{ME}^\star$, presented in Fig. 3.1 can be proved by Lemma 3.32. We will prove only **(unf₁)** and drop the proofs of the other two, since they can be performed analogously. Let $p \doteq (\mathsf{init}\, z \leftarrow s\, \mathsf{in}\, t^\star)$ and $q \doteq s + \mathrm{do}\ z \leftarrow (\mathsf{init}\, z \leftarrow s\, \mathsf{in}\, t^\star); t$. Note that by Lemma 3.24, a either does not depend on v or $a \doteq v$. The former case is trivial and we stick to the latter one. By Lemma 3.32:

$$
\begin{aligned}
\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) &\simeq_\star \textstyle\bigcup_i \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathsf{init}\, z \leftarrow s\, \mathsf{in}\, t^i); r),\\
\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r) &\simeq_\star \textstyle\bigcup_i \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow (\mathsf{init}\, z \leftarrow s\, \mathsf{in}\, t^i); y \leftarrow t; r) \cup\\
&\qquad \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow s; r).
\end{aligned}
$$

Now the proof of (3.28) follows from (3.21):

$$\begin{aligned}
&\textstyle\bigcup_i \mathsf{NF}(\,\mathrm{do}\;\, x \leftarrow u; y \leftarrow (\mathrm{init}\, z \leftarrow s\,\mathrm{in}\, t^i); r)\\
&\quad\simeq_\star \textstyle\bigcup_{i>0} \mathsf{NF}(\mathrm{do}\;\, x \leftarrow u; y \leftarrow (\mathrm{init}\, z \leftarrow s\,\mathrm{in}\, t^i); r) \cup\\
&\qquad \textstyle\bigcup_{i=0} \mathsf{NF}(\mathrm{do}\;\, x \leftarrow u; y \leftarrow s; r)\\
&\quad\simeq_\star \textstyle\bigcup_i \mathsf{NF}(\mathrm{do}\;\, x \leftarrow u; z \leftarrow (\mathrm{init}\, z \leftarrow s\,\mathrm{in}\, t^i); y \leftarrow t; r) \cup\\
&\qquad \mathsf{NF}(\mathrm{do}\;\, x \leftarrow u; y \leftarrow s; r).
\end{aligned}$$

We have thus proved (3.27) for all axioms of $\mathsf{ME}^\star$. Let us proceed with the derivation rules. Verification of **(sym)** and **(trans)** is trivial due to symmetry and transitivity of $\simeq$. Let us verify the congruence rules.

Consider the congruence rule for signature symbols. Let for some $(s = t) \in \Phi$, $f \in \Sigma$, $p \doteq f(s)$ and $q \doteq f(t)$. By the premise of the lemma, for every $u_1, \ldots, u_n, r$ and every normal $a \in \mathcal{A}_\Sigma$:

$$\mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow b[s/z]; r) \simeq \mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow b[t/z]; r)$$

where $b := \mathsf{nf}(a[f(z)/v])$ and z is chosen so that $z \notin \mathrm{Vars}(a)$ from which we conclude with the help of (3.20) the goal:

$$\mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow a[f(s)/v]; r) \simeq \mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow a[f(t)/v]; r).$$

In the same way one can prove **(cong_fst)** and **(cong_snd)**. Let us consider **(cong_pair)**. Suppose for some s_1, s_2, t_1, t_2, $\Phi \vdash s_1 = t_1$, $\Phi \vdash s_2 = t_2$, $p \doteq \langle s_1, t_1\rangle$ and $q \doteq \langle s_2, t_2\rangle$. By normality and for typing reasons the atomic program a in (3.27) must be in either of the forms: $h_1(\ldots(h_k(v))\ldots)$ or $h_1(\ldots(h_k(f(w)))\ldots)$ where $f \in \Sigma$ and for every i, $h_i \in \{\mathrm{fst}, \mathrm{snd}\}$. Consider the former case. For typing reasons k must be at least 1. Suppose e.g. $h_k = \mathrm{fst}$ (case $h_k = \mathrm{snd}$ is analogous). Then the proof of (3.29) is as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\;\, \bar{x}\; \leftarrow \bar{u}; y \leftarrow a[\langle s_1, t_1\rangle/v]; r)\\
&\quad\simeq_\star \mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow h_1(\ldots(h_{k-1}(s_1))\ldots); r) && \text{[by 3.20]}\\
&\quad\simeq\;\; \mathsf{NF}(\mathrm{do}\;\, \bar{x} \leftarrow \bar{u}; y \leftarrow h_1(\ldots(h_{k-1}(s_2))\ldots); r) && \text{[by 3.26]}\\
&\quad\simeq_\star \mathsf{NF}(\mathrm{do}\;\, \bar{x}\; \leftarrow \bar{u}; y \leftarrow a[\langle s_2, t_2\rangle/v]; r) && \text{[by 3.20].}
\end{aligned}$$

Suppose $a \doteq h_1(\ldots(h_k(f(w)))\ldots)$. Let v_1, v_2 be two fresh variables such that $\langle v_1, v_2\rangle$ has the same return type as v. By Lemma 3.22, $\mathsf{nf}(a[\langle s_1, v_2\rangle/v])$, which is equal to $h_1(\ldots(h_k(f(\mathsf{nf}(w[\langle s_1, v_2\rangle/v]))))\ldots)$, is atomic. For the same reason, $\mathsf{nf}(a[\langle v_1, t_2\rangle/v])$ must be atomic. Then the proof of (3.29) runs as follows:

$$\mathsf{NF}(\mathrm{do}\;\, \bar{x}\; \leftarrow \bar{u}; y \leftarrow a[\langle s_1, t_1\rangle/v]; r)$$

$$
\begin{aligned}
&\simeq_\star \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[\langle s_1, v_2\rangle / v])[t_1/v_2]; r) && \text{[by 3.20]}\\
&\simeq\ \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[\langle s_1, v_2\rangle / v])[t_2/v_2]; r) && \text{[by 3.26]}\\
&\simeq_\star \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[\langle v_1, t_2\rangle / v])[s_1/v_1]; r) && \text{[by 3.20]}\\
&\simeq\ \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathsf{nf}(a[\langle v_1, t_2\rangle / v])[s_2/v_1]; r) && \text{[by 3.26]}\\
&\simeq_\star \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[\langle s_2, t_2\rangle / v]; r). && \text{[by 3.20]}
\end{aligned}
$$

We proceed with the verification of the remaining congruence rules capturing the control operators. Note that for all of them by Lemma 3.24, the atomic program a in (3.27) must be equal to v.

Consider the congruence rule for ret. Suppose $p \doteq \mathrm{ret}\, s$, $q \doteq \mathrm{ret}\, t$ and $\vdash_{\mathsf{ME}^\star} s = t$. By (3.20) the goal (3.27) is equivalent to

$$\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathrm{ret}\, s; \mathsf{nf}(r)) \simeq \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathrm{ret}\, t; \mathsf{nf}(r)).$$

By Remark 3.23, $\mathsf{nf}(r)$ has the form $\sum_j \mathsf{do}\ \bar{z}^j \leftarrow \bar{a}^j; w_j$. Since by (3.20),

$$
\begin{aligned}
\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathrm{ret}\, s; \mathsf{nf}(r)) &\simeq_\star \textstyle\bigcup_j \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[s/y]; w_j[s/y]),\\
\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow \mathrm{ret}\, t; \mathsf{nf}(r)) &\simeq_\star \textstyle\bigcup_j \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[t/y]; w_j[t/y])
\end{aligned}
$$

we will be done once we prove the equivalence

$$\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[s/y]; w_j[s/y]) \simeq \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[t/y]; w_j[t/y]) \tag{3.29}$$

for all j. By Lemmas 3.27 and 3.26, the latter equivalence becomes trivial if either of the a^j_i, w_j has a Kleene star on top. Therefore we assume henceforth that all the a^j_i are atomic and every w_j is either atomic or is of the form $\mathrm{ret}\, u$. Let us fix j and suppose $w_j \doteq \mathrm{ret}\, u$. We proceed by further induction over $m := |\bar{a}|$. Let $m = 0$. Recall that $(s = t) \in \Phi$. Therefore, $\Phi \vdash_{\mathsf{ME}^\star} u[s/y] = u[t/y]$ and we are done by Lemma 3.33. Let $m > 0$. Then the proof of (3.29) is as follows:

$$
\begin{aligned}
&\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{x} \leftarrow \bar{a}[s/y]; \mathrm{ret}\, u[s/y])\\
&\quad = \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x_1 \leftarrow a_1[s/y]; \ldots; x_m \leftarrow a_m[s/y]; \mathrm{ret}\, u[s/y])\\
&\quad \simeq \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x_1 \leftarrow a_1[t/y]; \ldots; x_m \leftarrow a_m[s/y]; \mathrm{ret}\, u[s/y]) && \text{[by 3.26]}\\
&\quad \simeq \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x_1 \leftarrow a_1[t/y]; \ldots; x_m \leftarrow a_m[t/y]; \mathrm{ret}\, u[t/y]) && \text{[by ind.]}\\
&\quad = \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{x} \leftarrow \bar{a}[t/y]; \mathrm{ret}\, u[t/y]).
\end{aligned}
$$

The situation when w_j is atomic can be reduced to the one just considered by the equivalence:

$$
\begin{aligned}
&\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[u/y]; w_j[u/y])\\
&\qquad\qquad \simeq_\star \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; \bar{z}^j \leftarrow \bar{a}^j[u/y]; z \leftarrow w_j[u/y]; \mathrm{ret}\, z),
\end{aligned}
$$

which itself follows from (3.20). We have thus completed the proof of **(cong_ret)**.

Let us prove the congruence rule for binding. Suppose $p \doteq (\mathrm{do}\ z \leftarrow s_1; t_1)$, $q \doteq (\mathrm{do}\ z \leftarrow s_2; t_2)$ and $\{s_1 = s_2, t_1 = t_2\} \subseteq \Phi$. The proof of (3.27) runs as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) && \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow s_1; y \leftarrow t_1; r) && \text{[by 3.20]}\\
&\quad \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow s_2; y \leftarrow t_1; r) && \text{[by 3.26]}\\
&\quad \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow s_2; y \leftarrow t_2; r) && \text{[by 3.26]}\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ x \leftarrow u; y \leftarrow a[q/v]; r). && \text{[by 3.20]}
\end{aligned}$$

Let us prove the congruence rule for the Kleene star. Let $p \doteq (\mathrm{init}\, z \leftarrow s_1 \,\mathrm{in}\, t_1^\star)$, $q \doteq (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^\star)$ and $\{s_1 = s_2, t_1 = t_2\} \subseteq \Phi$. By Lemma 3.32:

$$\begin{aligned}
\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) &\simeq_\star \bigcup\nolimits_i \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_1 \,\mathrm{in}\, t_1^i); r),\\
\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r) &\simeq_\star \bigcup\nolimits_i \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^i); r).
\end{aligned}$$

Therefore, it suffices to prove for every i the equivalence:

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_1 \,\mathrm{in}\, t_1^i); r) \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^i); r).$$

We proceed by further induction over i. If $i = 0$ we are immediately done by (3.26). If $i > 0$ then the proof is as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_1 \,\mathrm{in}\, t_1^i); r) && \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow (\mathrm{init}\, z \leftarrow s_1 \,\mathrm{in}\, t_1^{i-1}); y \leftarrow t_1; r) && \text{[by 3.20]}\\
&\quad \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^{i-1}); y \leftarrow t_1; r) && \text{[by ind.]}\\
&\quad \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^{i-1}); y \leftarrow t_2; r) && \text{[by 3.26]}\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s_2 \,\mathrm{in}\, t_2^i); r). && \text{[by 3.20]}
\end{aligned}$$

Let us prove the congruence rule for the choice. Let $p \doteq s_1 + t_1$, $q \doteq s_2 + t_2$ and $\{s_1 = s_2, t_1 = t_2\}$. Then the proof runs as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) && \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (s_1 + t_1); r) && \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow s_1; r) \cup \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow t_1; r) && \text{[by def. of NF]}\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow s_2; r) \cup \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow t_2; r) && \text{[by 3.26]}\\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r). &&
\end{aligned}$$

We have thus verified all the congruence rules. Let us verify **(inst)**. Let $p \doteq s_1[t/v]$, $q \doteq s_2[t/v]$ and $s_1 = s_2 \in \Phi$. Then the proof is as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) \\
&\quad = \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[s_1[t/z]/v]; r) \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow \mathrm{ret}\, t; y \leftarrow a[s_1/v]; r) && \text{[by 3.20]} \\
&\quad \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; z \leftarrow \mathrm{ret}\, t; y \leftarrow a[s_2/v]; r) && \text{[by 3.26]} \\
&\quad \simeq_\star \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[s_2[t/z]/v]; r). && \text{[by 3.20]} \\
&\quad = \mathsf{NF}(\mathrm{do}\ x \leftarrow u; y \leftarrow a[q/v]; r).
\end{aligned}$$

Finally, we need to prove the induction rules **($\mathbf{ind_1}$)** and **($\mathbf{ind_2}$)**. The proofs easily follow from Lemma 3.32. We prove only **($\mathbf{ind_1}$)** and drop the analogous case of the other one. Suppose $p \doteq (\mathrm{init}\, z \leftarrow s \,\mathrm{in}\, t^\star) + s$, $q \doteq s$ and $(\mathrm{do}\ z \leftarrow s; t + s = s) \in \Phi$. By Lemma 3.24, $a \doteq v$ in (3.28). By Lemma 3.32 we are left to show that for every i:

$$\begin{aligned}
\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow s; r) \simeq\ & \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow s; r) \cup \\
& \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow (\mathrm{init}\, z \leftarrow s \,\mathrm{in}\, t^i); r).
\end{aligned}$$

This equivalence can be easily proved by induction over i similar to the case of the congruence rule for the Kleene star. □

After instantiating in Lemma 3.34 Φ with $\emptyset$, $\bar{u}$ with the empty sequence, r with $\mathrm{ret}\, y$, and $\simeq$ with $\simeq_\star$, we obtain

Corollary 3.35. *Let p and q be two programs with the same computational return type. Then $\vdash_{\mathsf{ME}^\star} p = q$ implies $\mathsf{NF}(p) \simeq_\star \mathsf{NF}(q)$.*

Lemma 3.36. *Let Φ be a set of program equations and let $\simeq$ be an equivalence relation over $\mathcal{S}_\Sigma$ such that $\simeq$ is weaker than $\simeq^\Phi_\omega$ and for every $(s = t) \in \Phi$, every programs $u_1, \dots, u_n, r$ and every normal $a \in \mathcal{A}_\Sigma$:*

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[s/v]; r) \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[t/v]; r). \tag{3.30}$$

Let p and q be two programs such that $\Phi \vdash_{\mathsf{ME}^\omega} p = q$. Then for every $u_1, \dots, u_n, r$ and for every normal $a \in \mathcal{A}_\Sigma$:

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[p/v]; r) \simeq \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[q/v]; r). \tag{3.31}$$

Proof. We will be done by induction over proof complexity of $\Phi \vdash_{\mathsf{ME}^\omega} p = q$ once we prove the two specific cases: $p = q$ is an axiom of ME^ω and $p = q$ is obtained from Φ in one step by one of the derivation rules. Since ME^ω share the axioms with $\mathsf{ME}^\star$ and $\simeq^\Phi_\omega$ is weaker than $\simeq^\Phi_\star$ the former statement follows from Lemma 3.34, with $\simeq$ instantiated by $\simeq^\Phi_\omega$. Let us consider the latter statement. For all the rules of ME^ω which happen to be rules of $\mathsf{ME}^\star$ we obtain the proof by calling Lemma 3.34 once more. The only remaining rule is **(ω)**. Let

$$\textbf{(}\boldsymbol{\omega}\textbf{)}\quad \frac{\forall i.\ \mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^i\,\mathsf{res}\, x \rightarrow l \leqslant w}{\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^\star\,\mathsf{res}\, x \rightarrow l \leqslant w}$$

be the instance of **(ω)** in question, i.e.

$$\{\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^i\,\mathsf{res}\, x \rightarrow l = \mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^i\,\mathsf{res}\, x \rightarrow l + w\}_i \subseteq \Phi, \tag{3.32}$$

$p \doteq (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^\star\,\mathsf{res}\, x \rightarrow l)$ and $q \doteq (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^\star\,\mathsf{res}\, x \rightarrow l) + w$. Note that by Lemma 3.24 a in (3.31) must be equal to v, and thus the goal (3.31) transforms to:

$$\begin{aligned}&\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x \leftarrow (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^\star); y \leftarrow l; r)\\ &\quad\simeq^\Phi_\omega \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x \leftarrow (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^\star); y \leftarrow l; r) \cup\\ &\quad\quad \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow w; r).\end{aligned}$$

By Lemma 3.32 we switch equivalently to

$$\begin{aligned}&\mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x \leftarrow (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^i); y \leftarrow l; r)\\ &\quad\simeq^\Phi_\omega \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; x \leftarrow (\mathsf{init}\, x \leftarrow s\,\mathsf{in}\, t^i); y \leftarrow l; r) \cup\\ &\quad\quad \mathsf{NF}(\mathsf{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow w; r).\end{aligned}$$

which should be proved for every i. In fact, due to (3.32), for every i the latter equivalence is an instance of the assumption (3.30). The proof is thus completed. □

After instantiating in Lemma 3.36 Φ with $\emptyset$, $\bar{u}$ with the empty sequence, r with $\mathsf{ret}\, y$ and $\simeq$ with $\simeq_\star$, we obtain

Corollary 3.37. *Let p and q be two programs with the same computational return type. Then $\vdash_{\mathsf{ME}^\omega} p = q$ implies $\mathsf{NF}(p) \simeq_\omega \mathsf{NF}(q)$.*

Lemma 3.38. *In any strong continuous Kleene monad, for every p with a computational return type: $p = \sup \mathsf{NF}(p)$.*

Proof. Let us fix for the sequel an underlying strong continuous Kleene monad $\mathbb{T}$. Observe that by the definition of NF, for every $t \in \mathsf{NF}(p)$, $t \leqslant p$. Let q be a program satisfying the condition: for every $t \in \mathsf{NF}(p)$, $t \leqslant q$. We will be done as soon as we prove that $p \leqslant q$.

W.l.o.g. we assume that p has the form $(\mathsf{do}\ x \leftarrow w; \mathsf{ret}\, x)$ which is justified by the first unit law. We proceed by induction over n, the number of Kleene stars occurring in p. Let $n = 0$ and let $\bar{p}$ be a vector of shallow-deterministic programs such that $\mathsf{nf}(p) \doteq \sum_i p_i$. Then $\mathsf{NF}(p) = \bigcup_i \mathsf{NF}(p_i) = \{p_1, \dots, p_m\}$. Therefore $p = \sum_i p_i \leqslant q$ and we are done.

Suppose $n > 0$. Let us normalise p under the associativity rule. Then it takes the form $p \doteq \sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i$ (in particular this sum might contain only one summand). We successively transform p by replacing every $(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i)$ where for some k, p^i_k has the form $u + r$ by the two new summands:

$$\mathrm{do}\ x^i_1 \leftarrow p^i_1; \ldots; x_{k-1} \leftarrow p^i_{k-1}; x_k \leftarrow u; x_{k+1} \leftarrow p^i_{k+1}; \ldots; \mathrm{ret}\, z_i,$$
$$\mathrm{do}\ x^i_1 \leftarrow p^i_1; \ldots; x_{k-1} \leftarrow p^i_{k-1}; x_k \leftarrow r; x_{k+1} \leftarrow p^i_{k+1}; \ldots; \mathrm{ret}\, z_i$$

normalised under associativity. We repeat this transformation as long as possible. Termination is guaranteed by Lemma 2.16. We have thus ensured that for every i and k, p^i_k is either atomic or deadlock or Kleene star. Note that by (3.20) all these manipulations do not change $\mathsf{NF}(p)$, i.e. the assumption $\forall t \in \mathsf{NF}(p).\, t \leqslant q$ is maintained. We have thus for every i: $\forall t \in \mathsf{NF}(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i).\, t \leqslant q$. We will be done if we manage to prove that for every i

$$\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i \leqslant q. \tag{3.33}$$

Indeed, then we would have $p = \sum_i \mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i \leqslant q$.

We proceed with the proof of (3.33). Let us fix some index i. If for every k p^i_k does not contain Kleene star, then the proof of (3.33) runs the same way as the proof of the induction base. Suppose, on the contrary, for some k, $p^i_k \doteq (\mathrm{init}\, x \leftarrow u\, \mathrm{in}\, r^\star)$. Then $\mathsf{NF}(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i) = \bigcup_j \mathsf{NF}(q_j)$ with

$$q_j := \mathrm{do}\ x^i_1 \leftarrow p^i_1; \ldots; x_{k-1} \leftarrow p^i_{k-1}; x_k \leftarrow (\mathrm{init}\, x \leftarrow u\, \mathrm{in}\, r^j); x_{k+1} \leftarrow p^i_{k+1}; \ldots; \mathrm{ret}\, z_i.$$

According to the assumption, for every $t \in \mathsf{NF}(q_j)$, $t \leqslant q$. It can easily be seen by construction that the number of Kleene stars in $(\mathrm{do}\ \bar{x}^i \leftarrow \bar{p}^i; \mathrm{ret}\, z_i)$ is not greater than n. Therefore, the number of Kleene stars in every q_j is strictly less than n. By the induction hypothesis, for every j, $q_j \leqslant q$. For every j we have thus by **(assoc)**:

$$\mathrm{init}\, x \leftarrow (\mathrm{do}\ x^i_1 \leftarrow p^i_1; \ldots; x_{k-1} \leftarrow p^i_{k-1}; u)\, \mathrm{in}\, r^j\, \mathrm{res}\, x_k \rightarrow (\mathrm{do}\ \ldots; \mathrm{ret}\, z_i) \leqslant q$$

from which we deduce by **(ω)**:

$$\mathrm{init}\, x \leftarrow (\mathrm{do}\ x^i_1 \leftarrow p^i_1; \ldots; x_{k-1} \leftarrow p^i_{k-1}; u)\, \mathrm{in}\, r^\star\, \mathrm{res}\, x_k \rightarrow (\mathrm{do}\ \ldots; \mathrm{ret}\, z_i) \leqslant q.$$

By **(init)** the letter inequality is equivalent to (3.33). □

By Corollary 3.37 we know that $\vdash_{\mathsf{ME}^\omega} p = q$ implies $\mathsf{NF}(p) \simeq_\omega \mathsf{NF}(q)$. On the other hand, if for some p, q, $\mathsf{NF}(p) \simeq_\omega \mathsf{NF}(q)$, then for every $t \in \mathsf{NF}(p)$ there is $s \in \mathsf{NF}(q)$ such that $\vdash_{\mathsf{ME}^\omega} s = t$ and thus $\vdash_{\mathsf{ME}^\omega} t = s \leqslant q$ which by Lemma 3.38 implies $\vdash_{\mathsf{ME}^\omega} p \leqslant q$. By the symmetric argument, $\vdash_{\mathsf{ME}^\omega} q \leqslant p$. We have thus proved

Corollary 3.39. *Let p and q be two programs with the same computational return type. Then $\vdash_{\mathsf{ME}^\omega} p = q$ iff $\mathsf{NF}(p) \simeq_\omega \mathsf{NF}(q)$.*

3.5 Undecidability and incompleteness

Whereas conditional program equality is undecidable even for the deterministic case (see Theorem 1.26), the unconditional case remains decidable not only for plain vanilla monads but also for additive monads (see Theorem 2.18). After introducing the Kleene star into the reasoning we made the underlying language look essentially more expressive, but the question is still whether MCE can indeed capture more interesting cases of program equivalence than those which can be captured either by ME or by Kleene algebra. The answer can be justified by an (un-)decidability theorem. E.g. it is a well-known fact, justified by Rice's theorem (cf. e.g. [Rog87]), that all the non-trivial properties of recursive programs (e.g. program equivalence) are undecidable. On the other hand, the language under consideration operates with a very restricted sort of recursion and there is no obvious way to model natural numbers so as to make this recursion full-fledged. Furthermore, program equivalence in Kleene algebra, a fair prototype of Kleene monads, is decidable [Koz94]. Despite these considerations in the present section, we establish two negative results: program equality in $\mathsf{ME}^\star$ is undecidable (Theorem 3.41) and program equality in ME^ω is non-r.e. (Theorem 3.42). The core of both proofs is a reduction from the Post Correspondence Problem (PCP) in which we are aided considerably by the theory of NF operator developed in Section 3.4.

Before we proceed with the announced proofs let us clarify the notion of PCP and the background terminology. Let Ξ be a finite alphabet, containing at least two distinct letters. Recall that Ξ^+ denotes the set of all nonempty words over Ξ. For every $t \in \Xi^+$ we denote by t^{-1} the reversed string. A *PCP instance* is a finite set of the form $\{(p_1, q_1), \dots, (p_n, q_n)\}$ where $\{p_1, q_1, \dots, p_n, q_n\} \subseteq \Xi^+$. A *solution* of a PCP instance $\{(p_1, q_1), \dots, (p_n, q_n)\}$ is the finite sequence of indices $i_1, \dots, i_n$ such that $p_{i_1} p_{i_2} \dots p_{i_n} = q_{i_1} q_{i_2} \dots q_{i_n}$.

A conventional way of presenting pairs of strings in PCP instances is by domino-like tables arranged vertically. Then a solution of a PCP instance can be presented by a table with two lines. E.g. the PCP instance $\left\{ \begin{array}{|c|}\hline 0 \\ \hline 100 \\ \hline\end{array}, \begin{array}{|c|}\hline 01 \\ \hline 00 \\ \hline\end{array}, \begin{array}{|c|}\hline 110 \\ \hline 11 \\ \hline\end{array} \right\}$ has as a solution

110	01	110	0
11	00	11	100

. The following theorem is the basis for the upcoming undecidability results.

Theorem 3.40 (cf. e.g. [RS97])**.** *It is undecidable to check if a PCP instance has a solution.*

Now we can proceed with the main text.

Theorem 3.41 (Undecidability)**.** *The equational theory of* $\mathsf{ME}^\star$ *is undecidable.*

Proof. Let $\Xi := \{a, b, c\}$ be an alphabet, and let $P := \{\langle p_1, q_1\rangle, \dots, \langle p_n, q_n\rangle\}$ be an arbitrary PCP instance in $\Xi\backslash\{c\}$. We can treat elements of Ξ as functional symbols whose types are $1 \to T1$. Then symbol concatenation will correspond to the binding operator. Given some $s_1, \dots, s_n \in \Xi$ we will thus use both notations $s_1 \dots s_n$ and $(\mathrm{do}\ s_1; \dots; s_n)$ interchangeably. The idea of the proof is to construct a program equation over the signature Ξ which is provable in $\mathsf{ME}^\star$ iff the PCP instance has a solution. Let

$$s := \mathrm{do}\ x \leftarrow \Big(\mathsf{init}\, x \leftarrow \textstyle\sum_{i=1}^{n} \mathsf{ret}(\mathrm{do}\ p_i; c; q_i^{-1})\ \mathsf{in} \sum_{i=1}^{n} \mathsf{ret}(\mathrm{do}\ p_i; x; q_i^{-1})^\star\Big); x.$$

It is easy to see that $\mathsf{NF}(s)$ is precisely the set of all strings of the form lcr^{-1} where $l = p_{i_1} \dots p_{i_k}$ and $r = q_{i_1} \dots q_{i_k}$ for some finite sequences of indices $i_1, \dots, i_k$. In the same way we define

$$r := \mathsf{init}\, x \leftarrow \mathsf{ret}\, c\ \mathsf{in} \textstyle\sum_{i=1}^{n} \mathsf{ret}(\mathrm{do}\ a; x; a + \mathrm{do}\ b; x; b)^\star$$

and observe that $\mathsf{NF}(r)$ is the set of all terms of the form $\mathsf{ret}(wcw^{-1})$. The fact that the PCP instance does have a solution can now be encoded by the inequality

$$\mathsf{ret}\, s \leqslant \mathrm{do}\ x \leftarrow r; \mathsf{ret}(s + x). \tag{3.34}$$

Let us show that if the inequality (3.34) is provable in $\mathsf{ME}^\star$ the PCP instance has a solution. Let (3.34) be provable. By Corollary 3.35, $\mathsf{NF}(\mathsf{ret}\, s) \lesssim_\star \mathsf{NF}(\mathrm{do}\ x \leftarrow r; \mathsf{ret}(s + x))$. By the definition of $\lesssim_\star$, there exists $w \in \Xi^+$ such that $\mathsf{NF}(s)$ is pointwise provably equal to $\mathsf{NF}(s) \cup \{wcw^{-1}\}$ or, equivalently, for some $w \in \Xi^+$, wcw^{-1} is provably equal to an element of $\mathsf{NF}(s)$. The latter precisely means that the PCP instance has a solution.

Now let us prove that if the PCP instance has a solution then (3.34) is provable in $\mathsf{ME}^\star$. If the PCP instance has a solution then by definition $\mathsf{NF}(s)$ contains at least one element of the form wcw^{-1} where $w \in \Xi^+$. By (3.21), $\mathsf{NF}(s) = \bigcup_i \gamma(\mathsf{unf}_i(s))$. It can easily be seen that for every i the words from $\gamma(\mathsf{unf}_{i+1}(s))\backslash\gamma(\mathsf{unf}_i(s))$, are longer than i. Therefore, wcw^{-1} belongs to some $\gamma(\mathsf{unf}_i(s))$, and thus the inequality $\vdash_{\mathsf{ME}^\star} wcw^{-1} \leqslant s$ can be shown by applying the axiom **(unf$_2$)** finitely many times. We have thus in $\mathsf{ME}^\star$:

$$\begin{aligned}
&\mathrm{do}\ x \leftarrow r; \mathsf{ret}(s + x)\\
&\qquad \geqslant \mathrm{do}\ x \leftarrow \mathsf{ret}(wcw^{-1}); \mathsf{ret}(s + x)\\
&\qquad = \mathsf{ret}(s + wcw^{-1}) = \mathsf{ret}\, s
\end{aligned}$$

which proves (3.34), and thus we are done. □

In contrast to strong Kleene monads, which are only undecidable, strong continuous Kleene monads are moreover logically intractable.

Theorem 3.42. *Program equality in* ME^ω *is non-r.e.*

Proof. We prove this theorem in much the same manner as Theorem 3.41. But instead of encoding PCP we encode its dual, which is co-r.e. complete. This encoding is inspired by [Koz96]. Again, let $\Xi = \{a, b, c\}$, and $\{\langle p_1, q_1\rangle, \dots, \langle p_n, q_n\rangle\}$ be an instance of PCP in the alphabet $\Xi\backslash\{c\}$. Besides the program s from Theorem 3.41 which generates all potential solutions of the PCP instance, we introduce a program t generating all pairs of distinct strings. The program t can be defined using the term r from the proof of Theorem 3.41 by the assignment:

$$\begin{aligned} t := \mathrm{do}\ x \leftarrow r; y \leftarrow \sharp; z \leftarrow \sharp;&\\ &\big(\mathrm{do}\ y; a; x; b; z + \mathrm{do}\ y; b; x; a; z +\\ &\quad \mathrm{do}\ x; b; z + \mathrm{do}\ x; a; z + \mathrm{do}\ y; a; x + \mathrm{do}\ y; b; x\big),\end{aligned}$$

where $\sharp$ denotes the 'chaos' program

$$\mathrm{init}\, x \leftarrow \mathrm{ret}\, \mathrm{ret} \star\ \mathrm{in}\big(\mathrm{ret}(\mathrm{do}\ x; a) + \mathrm{ret}(\mathrm{do}\ x; b)\big)^{\star}.$$

For example, it can be easily verified that $aacab \leqslant t$, but $bacab \nleqslant t$ which precisely captures the facts: $aa \neq (ab)^{-1}$ and $ba = (ab)^{-1}$. According to the definition of s, $\mathsf{NF}(s)$ presents the set of strings presenting the potential solutions of the PCP instance. In the same way, observe that $\mathsf{NF}(t)$ presents the set of pairs of distinct strings. We will be done with the proof once we show that $\vdash_{\mathsf{ME}^\omega} s \leqslant t$ is equivalent to $\mathsf{NF}(s) \subseteq \mathsf{NF}(t)$. By Corollary 3.39, $\vdash_{\mathsf{ME}^\omega} s \leqslant t \iff \mathsf{NF}(s) \lesssim_\omega \mathsf{NF}(t)$. But both $\mathsf{NF}(s)$ and $\mathsf{NF}(t)$ consist of deterministic ret-free programs. Therefore by definition, $\mathsf{NF}(s) \lesssim_\omega \mathsf{NF}(t)$ amounts to the inclusion $\mathsf{NF}(s) \subseteq \mathsf{NF}(t)$, and thus we are done. □

Remark 3.43. The result about strong continuous Kleene monads (scKM) we have established can be pushed a little bit forward by noticing that the encoding of PCP we have used does not make use of deadlock and the axioms for it. Careful examination of the proofs behind Corollary 3.39 on which the result of Theorem 3.42 is based shows that all the concerns about $\varnothing$ never affect the arguments. We can therefore introduce a class of monads whose complete axiomatisation is obtained from ME^ω by removing the rules for $\varnothing$, e.g. let us call these monads strong continuous pseudo-Kleene monads (scpKM), and obtain for them an analogue of Corollary 3.39. It can be seen then that application of the partiality monad transformer $\lambda X.\, X + 1$ to an scpKM produces an scKM, which means in particular that any program equation not involving $\varnothing$ is valid over the scpKM iff it is valid over scKM. As a consequence, for any class of monads $\mathcal{C}$ between scpKM and scKM, a program equation not involving $\varnothing$ is valid over $\mathcal{C}$ iff it is valid over scKM. Since the construction from Theorem 3.42 involves only programs that do not contain $\varnothing$, this results in the analogous non-r.e. property for $\mathcal{C}$.

One of the implications of these considerations is that, e.g. a powerset exception monad $\mathcal{P}(\_+E)$ that fails to be scKM but is nonetheless scpKM still cannot be captured by a reasonable tractable class of strong monads with nondeterminism and continuous iteration.

3.6 Contribution and related work

In the present chapter we have introduced the core concept of this thesis, which is the notion of strong Kleene monads. Our axiomatisation of Kleene monads was strongly influenced by Kozen's axiomatisation of Kleene algebra [Koz94]. Roughly, the calculus of strong Kleene monads can be seen as the union of Kleene algebra theory and the theory of strong monads. Unlike the more abstract concept of Kleene algebra, Kleene monads justify the computational interpretation of sequencing, Kleene stars, deadlock, etc. The most essential differences distinguishing Kleene monads from Kleene algebras are *multivariable contexts* and the *interpreted return operator*. These two features make MCE, an internal language of strong Kleene monads, very close to a full-fledged programming language. The non-recursive enumerability result (Theorem 3.42) formally justifies the fact that MCE is indeed a very expressive language unlike the components it is made of, i.e. ME and Kleene algebra, which are both decidable. Our proof of non-recursive enumerability is similar to an analogous proof for the Horn theory of $*$-continuous Kleene algebras [Koz96], but in our case already the equational theory fails to be r.e.

There is a number of works by other researchers attempting to capture the notion of monad-based recursion. A good many of them present attempts to generalise certain constructions from domain theory or process algebra originally established for some very specific types of side-effects and thus presenting substantial limitations for further generalisation over reasonably wide classes of computational effects, particularly those that include stateful computations. A widely recognized work of Crole and Pitts [CP90] introduces a notion of a *fixpoint object* as a basis for fixed-point computations with monads. A notable distinction between the approach of Crolle and Pitts and ours is that the calculus from [CP90] is only consistent with respect to domain-like models, whereas our calculus is also consistent with respect to rather simpler set-theoretic models. Another important distinction is that our notion of recursion essentially benefits from the explicit operations for nondeterminism, which are absent in [CP90] and which cannot be naturally introduced due to underspecified computational behaviour of nondeterministic choice. Finally, no counterpart of ω-continuity appears in [CP90], hence no completeness/decidability issues with respect to 'standard' models (the problem central for our studies) have been raised.

A line of research studying coalgebraic traces, e.g. [HJS06, Jac10], involves fixed points of effectful operators where the effects are supposed to be of a certain type, namely those that admit a coalgebraic interpretation. These are, e.g. the powerset monad, exception monad, etc. On the other hand, the result of instantiating the underlying monad with stateful computations is not easily interpreted.

A significant source of inspiration for doing effectful recursion is the notion of *iteration theory* [BE93]. Iteration theories are sum and substance of algebraic recursion; they capture the identities featured by recursion operators and known to be complete over a wide range of various interpretations, including order-enriched ones. The notion of the *iteration monad* introduced in [AMV10] can be seen as an interpretation of iteration theories in a Kleisli category of a monad. It should be noted that the idea of introducing recursion for effectful computations by instantiating an existing algebraic notion of recursion in a Kleisli category is generally coherent with our approach. Roughly, iteration theories are to iteration monads as Kleene algebras are to Kleene monads. On the other hand, we should emphasise that in contrast to others we do take multivariable contexts seriously, i.e. our treatment of monad-based recursion implies all the axioms of monadic strength, plus we adhere to all the monad laws. Theorem 3.42 justifies the fact that this attitude dramatically contrasts the existing approaches based on iterative theories, as the latter commonly lead to expressly positive results [SP00, HK02b].

A notable extension of Kleene algebra pretty much resembling the look of the metalanguage of control and effects has been introduced in [AHK07]. No essential theoretical study of the corresponding logical calculus has been undertaken though.

The notion of a complete additive monad previously appears in [EG03] but under the name Kleene monad and thus creates a clash with our terms. Consequently, our notion of a Kleene monad is weaker than the one from [EG03]. While sharing the concepts (and terms) with [EG03] there does not seem to be any essential overlap in contribution.

Chapter 4

Decidable fragments of MCE

An immediate consequence of Theorem 3.42 is incompleteness of the calculus $\mathsf{ME}^\star$ over strong continuous Kleene monads. Furthermore, it cannot be completed in any finitary way. Although our encoding of Post Correspondence Problem makes use of only a small part of the expressive power of the language, one may still hope to impose some reasonable syntactic restrictions over programs in order to regain completeness. A natural restriction of this kind is to limit the usage of the return operator. In this chapter we introduce various solvable classes of the program equality problem, attempting to come as close as possible to the non-r.e. border.

Throughout the chapter we assume that the underlying signature Σ is plain.

4.1 Computational networks and the commutation lemma

According to the soundness and completeness result (Theorem 3.12), all the properties of strong Kleene monads are encoded in the calculus $\mathsf{ME}^\star$. Being itself rather simple and concise, the proof calculus $\mathsf{ME}^\star$ entails a great deal of facts and proof principles that are far from being immediately apparent.

In Section 3.2 we introduced a pattern-matching notation as a shorthand for the commonly used program idioms:

$$\begin{array}{rcl} \mathrm{do}\ \bar{x} \leftarrow p; q & \text{for} & \mathrm{do}\ z \leftarrow p; q[\mathrm{pr}^n_1(z)/x_1, \dots, \mathrm{pr}^n_n(z)/x_n] \quad \text{and} \\ \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star & \text{for} & \mathrm{init}\,z \leftarrow p\,\mathrm{in}\,q[\mathrm{pr}^n_1(z)/x_1, \dots, \mathrm{pr}^n_n(z)/x_n]^\star. \end{array}$$

In most cases we do not need to distinguish programs in pattern-matching notation from the ordinary ones. In case we do, we refer to the former ones as *pattern-matching* *programs*. Most of the statements about ordinary programs stay valid for pattern-matching programs. E.g. the modified version of axiom **(dist$_1^+$)**

$$\mathrm{do}\ \bar{x} \leftarrow (p+q); r = \mathrm{do}\ \bar{x} \leftarrow p; r + \mathrm{do}\ \bar{x} \leftarrow q; r$$

is clearly a provable identity. Non-trivial examples important for the further calculations are the pattern-matching versions of the laws **(assoc)** and **(init)**.

Lemma 4.1. *Let $\bar{x}$ and $\bar{y}$ be two vectors of variables, satisfying the condition:* $\mathrm{Vars}(r) \cap \mathrm{Vars}(\bar{y}) \subseteq \mathrm{Vars}(\bar{x})$. *Then the identities*

$$\mathrm{do}\ \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow p; q); r = \mathrm{do}\ \bar{y} \leftarrow p; \bar{x} \leftarrow q; r, \tag{4.1}$$

$$\mathrm{init}\, \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow p; q)\, \mathrm{in}\, r^\star = \mathrm{do}\ \bar{y} \leftarrow p; \mathrm{init}\, \bar{x} \leftarrow q\, \mathrm{in}\, r^\star \tag{4.2}$$

are provable in ME*.

Proof. Let us prove the first identity. We have:

$$\begin{aligned}
&\mathrm{do}\ \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow p; q); r \\
&\quad = \mathrm{do}\ z \leftarrow (\mathrm{do}\ v \leftarrow p; q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m]); \\
&\qquad r[\mathrm{pr}_1^n(z)/x_1, \dots, \mathrm{pr}_n^n(z)/x_n] && \text{[by def.]} \\
&\quad = \mathrm{do}\ v \leftarrow p; z \leftarrow q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m]; \\
&\qquad r[\mathrm{pr}_1^n(z)/x_1, \dots, \mathrm{pr}_n^n(z)/x_n] && \text{[by \textbf{(assoc)}]} \\
&\quad = \mathrm{do}\ v \leftarrow p; \bar{x} \leftarrow q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m]; r && \text{[by def.]} \\
&\quad = \mathrm{do}\ v \leftarrow p; (\mathrm{do}\ \bar{x} \leftarrow q; r)[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m] && \text{[by assm.]} \\
&\quad = \mathrm{do}\ \bar{y} \leftarrow p; (\mathrm{do}\ \bar{x} \leftarrow q; r). && \text{[by def.]}
\end{aligned}$$

In a similar fashion:

$$\begin{aligned}
&\mathrm{init}\, \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow p; q)\, \mathrm{in}\, r^\star \\
&\quad = \mathrm{init}\, z \leftarrow (\mathrm{do}\ v \leftarrow p; q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m])\, \mathrm{in} \\
&\qquad r[\mathrm{pr}_1^n(z)/x_1, \dots, \mathrm{pr}_n^n(z)/x_n]^\star && \text{[by def.]} \\
&\quad = \mathrm{do}\ v \leftarrow p; \mathrm{init}\, z \leftarrow q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m]\, \mathrm{in} \\
&\qquad r[\mathrm{pr}_1^n(z)/x_1, \dots, \mathrm{pr}_n^n(z)/x_n]^\star && \text{[by \textbf{(init)}]} \\
&\quad = \mathrm{do}\ v \leftarrow p; \mathrm{init}\, \bar{x} \leftarrow q[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m]\, \mathrm{in}\, r^\star && \text{[by def.]} \\
&\quad = \mathrm{do}\ v \leftarrow p; (\mathrm{init}\, \bar{x} \leftarrow q\, \mathrm{in}\, r^\star)[\mathrm{pr}_1^m(v)/y_1, \dots, \mathrm{pr}_m^m(v)/y_m] && \text{[by assm.]} \\
&\quad = \mathrm{do}\ \bar{y} \leftarrow p; \mathrm{init}\, \bar{x} \leftarrow q\, \mathrm{in}\, r^\star && \text{[by def.]}
\end{aligned}$$

and thus the second identity is proved. $\square$

In the following lemma we will collect some provable statements of $\mathsf{ME}^\star$ needed exceptionally for further references.

Lemma 4.2. *The following equations are provable in* $\mathsf{ME}^\star$*.*

$$\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star = \mathrm{init}\,\bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star)\,\mathrm{in}\,q^\star, \tag{4.3}$$

$$\begin{aligned}\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star = {} & \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star + \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star \\ & \mathrm{res}\,\bar{x} \to \big(\mathrm{init}\,\bar{x} \leftarrow r\,\mathrm{in}(q+r)^\star\big).\end{aligned} \tag{4.4}$$

Proof. Let us prove (4.3) by establishing mutual inequality. For one thing we have:

$$\begin{aligned}
&\mathrm{init}\,\bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star)\,\mathrm{in}\,q^\star \\
&\quad = \mathrm{init}\,\bar{x} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star); \mathrm{ret}\,\bar{x})\,\mathrm{in}\,q^\star && [\text{by } \textbf{(unit}_1\textbf{)}] \\
&\quad = \mathrm{do}\ \bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star); (\mathrm{init}\,\bar{x} \leftarrow \mathrm{ret}\,\bar{x}\,\mathrm{in}\,q^\star) && [\text{by } \textbf{(init)}] \\
&\quad \geqslant \mathrm{do}\ \bar{x} \leftarrow p; (\mathrm{init}\,\bar{x} \leftarrow \mathrm{ret}\,\bar{x}\,\mathrm{in}\,q^\star) && [\text{by } \textbf{(unf}_2\textbf{)}] \\
&\quad = \mathrm{init}\,\bar{x} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow p; \mathrm{ret}\,\bar{x})\,\mathrm{in}\,q^\star && [\text{by } \textbf{(init)}] \\
&\quad = \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star. && [\text{by } \textbf{(unit}_1\textbf{)}]
\end{aligned}$$

In order to prove the converse inequality observe that by **(unf$_1$)**:

$$\mathrm{do}\ \bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star); q \ \leqslant\ \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star.$$

Now the inequality in question follows by **(ind$_1$)**.

Let us prove (4.4) by establishing mutual equality. To that end we make use of the inequalities:

$$\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star \leqslant \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star, \tag{4.5}$$

$$\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,r^\star \leqslant \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star. \tag{4.6}$$

E.g. let us prove the first one. Observe that by **(unf$_1$)**, and **(dist$_2^+$)**:

$$\mathrm{do}\ \bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star); q \ \leqslant\ \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star$$

from which we conclude by **(ind$_1$)**:

$$\mathrm{init}\,\bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star)\,\mathrm{in}\,q^\star \ \leqslant\ \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star.$$

The left-hand side of the latter inequality is obviously greater than $(\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star)$ and hence (4.5). Let us proceed with the proof of (4.4). We have:

$$\begin{aligned}
&\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star \\
&\quad = \mathrm{init}\,\bar{x} \leftarrow \big(\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star\big)\,\mathrm{in}(q+r)^\star
\end{aligned}$$

$$\begin{aligned}
&= \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star + \\
&\quad \mathrm{init}\,\bar{x} \leftarrow \big(\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star\,\mathrm{res}\,\bar{x} \leftarrow (q+r)\big)\,\mathrm{in}(q+r)^\star && \text{[by 4.3]}\\
&\geqslant \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star + \\
&\quad \mathrm{init}\,\bar{x} \leftarrow (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \leftarrow r)\,\mathrm{in}(q+r)^\star && \text{[by 4.5, 4.6]}\\
&= \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star + \\
&\quad \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \to \big(\mathrm{init}\,\bar{x} \leftarrow r\,\mathrm{in}(q+r)^\star\big). && \text{[by Lem. 4.1]}
\end{aligned}$$

In order to complete the proof of (4.4) we are left to establish the converse inequality. Let us denote the right-hand side of (4.3) by w and observe the following:

$$\begin{aligned}
w &\doteq \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star + \\
&\quad \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \to \big(\mathrm{init}\,\bar{x} \leftarrow r\,\mathrm{in}(q+r)^\star\big)\\
&= p + (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \to q) + \\
&\quad (\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \to r) + \\
&\quad \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star \\
&\quad \mathrm{res}\,\bar{x} \to \big(\mathrm{init}\,\bar{x} \leftarrow r\,\mathrm{in}(q+r)^\star\,\mathrm{res}\,\bar{x} \to (q+r)\big) && \text{[by } \mathbf{(unf_1)}\text{]}\\
&\geqslant \mathrm{do}\;\; \bar{x} \leftarrow \big(\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star + && \text{[by Lem. 4.1,}\\
&\quad \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^\star\,\mathrm{res}\,\bar{x} \to (\mathrm{init}\,\bar{x} \leftarrow r\,\mathrm{in}(q+r)^\star)\big); (q+r) && \mathbf{(dist_2^+)}\text{]}\\
&= \mathrm{do}\;\; \bar{x} \leftarrow w; (q+r).
\end{aligned}$$

By $\mathbf{(ind_1)}$ we conclude $w \geqslant \mathrm{init}\,\bar{x} \leftarrow w\,\mathrm{in}(q+r)^\star$. Since w is evidently greater than p, $\mathrm{init}\,\bar{x} \leftarrow w\,\mathrm{in}(q+r)^\star \geqslant \mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}(q+r)^\star$ and thus we are done. □

Definition 4.3 (Computational net)**.** Let $\bar{x}^1, \ldots, \bar{x}^n$ be a collection of vectors of distinct variables; let for every i, j such that $1 \leqslant i, j \leqslant n$, $p_{i,j}$ be a pattern-matching program whose return type is the same as the type of $\bar{x}^j$; and let for every i, j, k, $\mathrm{Vars}(p_{i,j}) \cap \mathrm{Vars}(\bar{x}^k) \subseteq \mathrm{Vars}(\bar{x}^i)$. We call the triple $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$ a *computational net*.

Let $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$ be a computational net. For every $0 \leqslant m \leqslant n$ we recursively define the programs $p^m_{i,j}$ as follows:

- $p^0_{i,i} := \mathrm{ret}\,\bar{x}^i + p_{i,i}$,
- $p^0_{i,j} := p_{i,j}$, if $i \neq j$,
- $p^m_{i,j} := p^{m-1}_{i,j} + \mathrm{init}\,\bar{x}^m \leftarrow p^{m-1}_{i,m}\,\mathrm{in}(p^{m-1}_{m,m})^\star\,\mathrm{res}\,\bar{x}^m \to p^{m-1}_{m,j}$.

Let for every appropriate i, j, $\hat{p}_{i,j} := p^n_{i,j}$. We call the triple $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, \hat{p}_{i,j})$ the *transitive closure* of $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$. Let us show that $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, \hat{p}_{i,j})$ is indeed a computational net, i.e. let us establish the inclusion $\mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(\hat{p}_{i,j}) \subseteq \mathrm{Vars}(\bar{x}^i)$ for every i, j, k. Let us show that moreover $\mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^m_{i,j}) \subseteq \mathrm{Vars}(\bar{x}^i)$ by induction over m. Case $m = 0$ is trivial. If $m > 0$ then

$$\begin{aligned}
&\mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^m_{i,j}) \\
&\quad = \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{i,j}) \cup \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{i,m}) \cup \\
&\quad\quad \mathrm{Vars}(\bar{x}^k) \cap (\mathrm{Vars}(p^{m-1}_{m,m}) \backslash \mathrm{Vars}(\bar{x}^m)) \cup \\
&\quad\quad \mathrm{Vars}(\bar{x}^k) \cap (\mathrm{Vars}(p^{m-1}_{m,j}) \backslash \mathrm{Vars}(\bar{x}^m)) && [\text{by def. of } p^m_{i,j}] \\
&\quad \subseteq \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{i,j}) \cup \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{i,m}) \cup \\
&\quad\quad \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{m,m}) \cup \mathrm{Vars}(\bar{x}^k) \cap \mathrm{Vars}(p^{m-1}_{m,j}) \\
&\quad \subseteq \mathrm{Vars}(\bar{x}^i) && [\text{by ind.}]
\end{aligned}$$

and we are done.

Lemma 4.4. *Let $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$ be a computational net and let $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, \hat{p}_{i,j})$ be the transitive closure of it. Then for all appropriate i, j, k:*

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{x}^k \leftarrow \hat{p}_{i,k}; \hat{p}_{k,j} \leqslant \hat{p}_{i,j}$$

Proof. More generally, we prove that provided $1 \leqslant i, j, l \leqslant n$ and $1 \leqslant k \leqslant l$,

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{x}^k \leftarrow p^l_{i,k}; p^l_{k,j} \leqslant p^l_{i,j}. \tag{4.7}$$

The proof is by induction over l. If $k = l$ we have:

$$\begin{aligned}
&\mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^l_{l,j} \\
&\quad = \mathrm{do}\ \bar{x}^l \leftarrow (p^{l-1}_{i,l} + \mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\ \bar{x}^l \to p^{l-1}_{l,l}); && [\text{by def. of} \\
&\quad\quad\quad (p^{l-1}_{l,j} + \mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{l,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\ \bar{x}^l \to p^{l-1}_{l,j}) && p^l_{i,l}, p^l_{l,j}] \\
&\quad = \mathrm{do}\ \bar{x}^l \leftarrow (\mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star); \\
&\quad\quad\quad (p^{l-1}_{l,j} + \mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{l,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\ \bar{x}^l \to p^{l-1}_{l,j}) && [\text{by } \mathbf{(unf_1)}] \\
&\quad = \mathrm{do}\ \bar{x}^l \leftarrow \big(\mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star + \\
&\quad\quad\quad \mathrm{init}\ \bar{x}^l \leftarrow (\mathrm{do}\ \bar{x}^l \leftarrow (\mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in} \\
&\quad\quad\quad (p^{l-1}_{l,l})^\star); p^{l-1}_{l,l})\ \mathrm{in}(p^{l-1}_{l,l})^\star); p^{l-1}_{l,j} && [\text{by } \mathbf{(dist^+_1)}] \\
&\quad = \mathrm{do}\ \bar{x}^l \leftarrow (\mathrm{init}\ \bar{x}^l \leftarrow (\mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star)\ \mathrm{in}(p^{l-1}_{l,l})^\star); p^{l-1}_{l,j} && [\text{by } \mathbf{(unf_2)}] \\
&\quad = \mathrm{init}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\ \bar{x}^l \to p^{l-1}_{l,j} && [\text{by } 4.3]
\end{aligned}$$

The last program of this calculation is by definition smaller than $p^l_{i,j}$. We have thus proved the induction base ($k = l = 1$) for (4.7) and partly the induction step. We will be done as soon as we prove the induction step for $k < l$. First, observe that

$$p^l_{i,k} \leqslant p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k}, \tag{4.8}$$

$$p^l_{k,j} \leqslant p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j}. \tag{4.9}$$

Indeed,

$$\begin{aligned}
p^l_{i,k} &= p^{l-1}_{i,k} + \mathrm{init}\,\bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\,\bar{x}^l \to p^{l-1}_{l,k} && [\text{by def. of } p^l_{i,k}]\\
&= p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}; p^{l-1}_{l,k} + \\
&\quad\ \mathrm{do}\ \bar{x}^l \leftarrow (\mathrm{init}\,\bar{x}^l \leftarrow p^{l-1}_{i,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\,\bar{x}^l \to p^{l-1}_{l,l}); p^{l-1}_{l,k} && [\text{by } \textbf{(unf}_1\textbf{)}]\\
&\leqslant p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}; p^{l-1}_{l,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k} && [\text{by def. of } p^l_{i,l}]\\
&\leqslant p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k} && [\text{by def. of } p^l_{i,l}]
\end{aligned}$$

and analogously,

$$\begin{aligned}
p^l_{k,j} &= p^{l-1}_{k,j} + \mathrm{init}\,\bar{x}^l \leftarrow p^{l-1}_{k,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\,\bar{x}^l \to p^{l-1}_{l,j} && [\text{by def. of } p^l_{k,j}]\\
&= p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^{l-1}_{l,j} + \\
&\quad\ \mathrm{init}\,\bar{x}^l \leftarrow (\mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^{l-1}_{l,l})\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\,\bar{x}^l \to p^{l-1}_{l,j} && [\text{by } \textbf{(unf}_2\textbf{)}]\\
&= p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^{l-1}_{l,j} + \\
&\quad\ \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; \mathrm{init}\,\bar{x}^l \leftarrow p^{l-1}_{l,l}\ \mathrm{in}(p^{l-1}_{l,l})^\star\ \mathrm{res}\,\bar{x}^l \to p^{l-1}_{l,j} && [\text{by } \textbf{(init)}]\\
&\leqslant p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^{l-1}_{l,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j} && [\text{by def. of } p^l_{l,j}]\\
&\leqslant p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j} && [\text{by def. of } p^l_{l,j}]
\end{aligned}$$

Now we can prove the induction step for (4.7) with $k < l$ as follows:

$$\begin{aligned}
&\mathrm{do}\ \bar{x}^k \leftarrow p^l_{i,k}; p^l_{k,j}\\
&\quad\leqslant \mathrm{do}\ \bar{x}^k \leftarrow (p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k}); p^l_{k,j} && [\text{by } 4.8]\\
&\quad\leqslant \mathrm{do}\ \bar{x}^k \leftarrow (p^{l-1}_{i,k} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k});\\
&\qquad\qquad (p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j}) && [\text{by } 4.9]\\
&\quad= \mathrm{do}\ \bar{x}^k \leftarrow p^{l-1}_{i,k}; p^{l-1}_{k,j} + \mathrm{do}\ \bar{x}^k \leftarrow p^{l-1}_{i,k}; \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j} +\\
&\qquad \mathrm{do}\ \bar{x}^k \leftarrow (\mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k}); \bar{x}^l \leftarrow p^{l-1}_{k,l}; p^l_{l,j} +\\
&\qquad \mathrm{do}\ \bar{x}^k \leftarrow (\mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,k}); p^{l-1}_{k,j} && [\text{by } \textbf{(dist}^+_2\textbf{)}]\\
&\quad\leqslant p^{l-1}_{i,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; \bar{x}^l \leftarrow p^{l-1}_{l,l}; p^l_{l,j} + && [\text{by ind.,}\\
&\qquad \mathrm{do}\ \bar{x}^l \leftarrow p^{l-1}_{i,l}; p^l_{l,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^{l-1}_{l,j} && \text{Lemma } 4.1]\\
&\quad\leqslant p^l_{i,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; \bar{x}^l \leftarrow p^l_{l,l}; p^l_{l,j} + && [\text{by def. of } p^l_{i,j},\\
&\qquad \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^l_{l,j} + \mathrm{do}\ \bar{x}^l \leftarrow p^l_{i,l}; p^l_{l,j} && p^l_{l,l}, p^l_{i,l}, p^l_{l,j}]
\end{aligned}$$

The latter term of this calculation is smaller than or equal to $p^l_{i,j}$. This follows from (4.7) with $k := l$, i.e. from the partial case of (4.7) which we had established previously. The proof of (4.7) is therefore complete. □

Lemma 4.5 (Commutation Lemma). *Let $(n, \lambda i.\,\bar{x}^i, \lambda i.\,\lambda j.\,p_{i,j})$, $(n, \lambda i.\,\bar{y}^i, \lambda i.\,\lambda j.\,q_{i,j})$ be two computation nets and let $(n, \lambda i.\,\bar{x}^i, \lambda i.\,\lambda j.\,\hat{p}_{i,j})$, $(n, \lambda i.\,\bar{y}^i, \lambda i.\,\lambda j.\,\hat{q}_{i,j})$ be their transitive closures. Suppose we can put into corespondence to every pair (i, j) such that $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant n$, a program $r_{i,j}$ so that for every appropriate i, j:*

$$\vdash_{\mathsf{ME}^\star} \textstyle\sum_k \mathrm{do}\ \bar{x}^k \leftarrow r_{i,k}; p_{k,j} = \sum_k \mathrm{do}\ \bar{y}^k \leftarrow q_{i,k}; r_{k,j}. \tag{4.10}$$

Suppose, moreover that for every i, j, k

$$\mathrm{Vars}(\bar{y}^i) \cap \mathrm{Vars}(p_{j,k}) \subseteq \mathrm{Vars}(\bar{x}^j), \qquad \mathrm{Vars}(\bar{y}^i) \cap \mathrm{Vars}(r_{j,k}) \subseteq \mathrm{Vars}(\bar{y}^j).$$

Then for every appropriate i, j:

$$\vdash_{\mathsf{ME}^\star} \textstyle\sum_k \mathrm{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} = \sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j}. \tag{4.11}$$

Proof. In order to prove the claim we show that the following inequalities are provable in $\mathsf{ME}^\star$ for all appropriate i, j, l, m which evidently implies the claim:

$$\textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j} \geqslant \mathrm{do}\ \bar{x}^l \leftarrow r_{i,l}; p^m_{l,j}, \tag{4.12}$$

$$\textstyle\sum_k \mathrm{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} \geqslant \mathrm{do}\ \bar{y}^l \leftarrow q^m_{i,l}; r_{l,j}. \tag{4.13}$$

Let us prove (4.12) by induction over m. If $m = 0$ then:

$$\begin{aligned}
&\textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j} \\
&\quad \geqslant \textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow q_{i,k}; r_{k,j} + r_{l,l} && \text{[by def. of } \hat{q}_{i,k}] \\
&\quad = \textstyle\sum_k \mathrm{do}\ \bar{x}^k \leftarrow r_{i,k}; p_{k,j} + r_{l,l} && \text{[by 4.10]} \\
&\quad \geqslant \mathrm{do}\ \bar{x}^l \leftarrow r_{i,l}; p_{l,j} + r_{l,l} \\
&\quad \geqslant \mathrm{do}\ \bar{x}^l \leftarrow r_{i,l}; p^0_{l,j}. && \text{[by def. of } p^0_{l,j}]
\end{aligned}$$

Let $m > 0$. First of all, let us observe the following:

$$\begin{aligned}
&\mathrm{do}\ \bar{x}^m \leftarrow \left(\textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m}\right); p^{m-1}_{m,m} \\
&\quad = \textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \bar{x}^m \leftarrow r_{k,m}; p^{m-1}_{m,m} && \text{[by Lemma 4.1]} \\
&\quad \leqslant \textstyle\sum_k \sum_{k'} \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \bar{y}^{k'} \leftarrow \hat{q}_{k,k'}; r_{k',m} && \text{[by ind.]} \\
&\quad = \textstyle\sum_k \sum_{k'} \mathrm{do}\ \bar{y}^{k'} \leftarrow (\mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \hat{q}_{k,k'}); r_{k',m} && \text{[by Lemma 4.1]} \\
&\quad \leqslant \textstyle\sum_{k'} \mathrm{do}\ \bar{y}^{k'} \leftarrow \hat{q}_{i,k'}; r_{k',m}. && \text{[by Lemma 4.4]}
\end{aligned}$$

If we now apply **(ind$_1$)** we obtain:

$$\vdash_{\mathsf{ME}^\star} \textstyle\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m} \geqslant \mathrm{init}\ \bar{x}^m \leftarrow \left(\sum_k \mathrm{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m}\right) \mathrm{in} (p^{m-1}_{m,m})^\star.$$

Since $\vdash_{\mathsf{ME}^\star} \sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m} \geqslant \mathsf{do}\ \bar{y}^i \leftarrow \hat{q}_{i,i}; r_{i,m} \geqslant r_{i,m}$ we can obtain:

$$\vdash_{\mathsf{ME}^\star} \textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m} \geqslant \mathsf{init}\ \bar{x}^m \leftarrow r_{i,m}\ \mathsf{in}(p^{m-1}_{m,m})^\star. \tag{4.14}$$

Let us return to the proof of (4.12). By the definition of $p^m_{l,j}$, (4.12) can be split into two inequalities:

$$\begin{aligned}
&\textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j} \geqslant \mathsf{do}\ \bar{x}^l \leftarrow r_{i,l}; p^{m-1}_{l,j},\\
&\textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j} \geqslant \mathsf{do}\ \bar{x}^l \leftarrow r_{i,l}; (\mathsf{init}\ \bar{x}^m \leftarrow p_{l,m}\ \mathsf{in}(p^{m-1}_{m,m})^\star\ \mathsf{res}\ \bar{x}^m \to p^{m-1}_{m,j}).
\end{aligned}$$

The first of them is precisely the induction hypothesis. The proof of the second runs as follows:

$$\begin{aligned}
&\textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j} \\
&\geqslant \textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \left(\sum_{k'} \mathsf{do}\ \bar{y}^{k'} \leftarrow \hat{q}_{i,k'}; \hat{q}_{k',k}\right); r_{k,j} && \text{[by Lem. 4.4]}\\
&= \textstyle\sum_k \sum_{k'} \mathsf{do}\ \bar{y}^{k'} \leftarrow \hat{q}_{i,k'}; \bar{y}^k \leftarrow \hat{q}_{k',k}; r_{k,j} && \text{[by Lem. 4.1]}\\
&\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{y}^{k'} \leftarrow \hat{q}_{i,k'}; \bar{x}^m \leftarrow r_{k',m}; p^{m-1}_{m,j} && \text{[by ind.]}\\
&\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{y}^{k'} \leftarrow \left(\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \hat{q}_{k,k'}\right); \bar{x}^m \leftarrow r_{k',m}; p^{m-1}_{m,j} && \text{[by Lem. 4.4]}\\
&\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{x}^m \leftarrow \left(\mathsf{do}\ \bar{y}^{k'} \leftarrow \left(\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \hat{q}_{k,k'}\right); r_{k',m}\right); p^{m-1}_{m,j} && \text{[by Lem. 4.1]}\\
&\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{x}^m \leftarrow \left(\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \bar{y}^{k'} \leftarrow \hat{q}_{k,k'}; r_{k',m}\right); p^{m-1}_{m,j} && \text{[by Lem. 4.1]}\\
&\geqslant \textstyle\sum_k \mathsf{do}\ \bar{x}^m \leftarrow (\mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; \mathsf{init}\ \bar{x}^m \leftarrow r_{k,m}\ \mathsf{in}(p^{m-1}_{m,m})^\star); p^{m-1}_{m,j} && \text{[by 4.14]}\\
&= \mathsf{init}\ \bar{x}^m \leftarrow \left(\textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,m}\right) \mathsf{in}(p^{m-1}_{m,m})^\star\ \mathsf{res}\ \bar{x}^m \to p^{m-1}_{m,j} && \text{[by Lem. 4.1]}\\
&\geqslant \mathsf{init}\ \bar{x}^m \leftarrow (\mathsf{do}\ \bar{x}^l \leftarrow r_{i,l}; p_{l,m})\ \mathsf{in}(p^{m-1}_{m,m})^\star\ \mathsf{res}\ \bar{x}^m \to p^{m-1}_{m,j} && \text{[by ind.]}\\
&= \mathsf{do}\ \bar{x}^l \leftarrow r_{i,l}; (\mathsf{init}\ \bar{x}^m \leftarrow p_{l,m}\ \mathsf{in}(p^{m-1}_{m,m})^\star\ \mathsf{res}\ \bar{x}^m \to p^{m-1}_{m,j}). && \text{[by Lem. 4.1]}
\end{aligned}$$

In a similar fashion we prove (4.13) by induction over m. For $m = 0$ we have:

$$\begin{aligned}
&\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j}\\
&\qquad\geqslant \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; p_{k,j} + r_{l,l} && \text{[by def. of } \hat{p}_{k,j}\text{]}\\
&\qquad= \textstyle\sum_k \mathsf{do}\ \bar{y}^k \leftarrow q_{i,k}; r_{k,j} + r_{l,l} && \text{[by 4.10]}\\
&\qquad\geqslant \mathsf{do}\ \bar{y}^l \leftarrow q_{i,l}; r_{l,j} + r_{l,l}\\
&\qquad\geqslant \mathsf{do}\ \bar{y}^l \leftarrow q^0_{i,l}; r_{l,j}. && \text{[by def. of } q^0_{i,l}\text{]}
\end{aligned}$$

Let $m > 0$. Again, we would like to prove an auxiliary statement, similar to (4.14). First observe that:

$$\begin{aligned}
&\mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{m,m}; \left(\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,i}\right)\\
&\qquad= \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow (\mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{m,m}; r_{m,k}); \hat{p}_{k,i} && \text{[by Lemma 4.1]}
\end{aligned}$$

$$\begin{aligned}
&\leqslant \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow \Big(\sum_{k'} \mathsf{do}\ \bar{x}^{k'} \leftarrow r_{m,k'}; \hat{p}_{k',k}\Big); \hat{p}_{k,i} && \text{[by ind.]}\\
&= \textstyle\sum_k \sum_{k'} \mathsf{do}\ \bar{x}^{k'} \leftarrow r_{m,k'}; \bar{x}^k \leftarrow \hat{p}_{k',k}; \hat{p}_{k,i} && \text{[by Lemma 4.1]}\\
&\leqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{x}^{k'} \leftarrow r_{m,k'}; \hat{p}_{k',i}. && \text{[by Lemma 4.4]}
\end{aligned}$$

Then by **(ind$_2$)**:

$$\begin{aligned}
\vdash_{\mathsf{ME}^\star} &\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,i} \geqslant \\
&\qquad \mathsf{init}\, \bar{y}^m \leftarrow \mathsf{ret}\, \bar{y}^m\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^m \to \Big(\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,i}\Big).
\end{aligned}$$

Since $\vdash_{\mathsf{ME}^\star} \sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,i} \geqslant \mathsf{do}\ \bar{x}^i \leftarrow r_{m,i}; \hat{p}_{i,i} \geqslant r_{m,i}$ we can obtain:

$$\vdash_{\mathsf{ME}^\star} \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,i} \geqslant \mathsf{init}\, \bar{y}^m \leftarrow \mathsf{ret}\, \bar{y}^m\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^m \to r_{m,i}. \tag{4.15}$$

By the definition of $p^m_{l,j}$, we will be done with the proof of (4.13) once we show that the following inequalities are provable in $\mathsf{ME}^\star$:

$$\begin{aligned}
&\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} \geqslant \mathsf{do}\ \bar{y}^l \leftarrow q^{m-1}_{i,l}; r_{l,j},\\
&\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} \geqslant \mathsf{do}\ \bar{y}^l \leftarrow (\mathsf{init}\, \bar{y}^m \leftarrow q_{i,m}\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^l \to q^{m-1}_{m,l}); r_{l,j}.
\end{aligned}$$

The first inequality coincides with the induction hypothesis. The second one is proven as follows:

$$\begin{aligned}
&\textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j}\\
&\quad\geqslant \textstyle\sum_k \sum_{k'} \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \bar{x}^{k'} \leftarrow \hat{p}_{k,k'}; \hat{p}_{k',j} && \text{[by Lem. 4.4]}\\
&\quad\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{x}^{k'} \leftarrow \Big(\sum_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,k'}\Big); \hat{p}_{k',j} && \text{[by Lem. 4.1]}\\
&\quad\geqslant \textstyle\sum_{k'} \mathsf{do}\ \bar{x}^{k'} \leftarrow (\mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{i,m}; r_{m,k'}); \hat{p}_{k',j} && \text{[by ind.]}\\
&\quad\geqslant \textstyle\sum_{k'} \sum_k \mathsf{do}\ \bar{x}^{k'} \leftarrow (\mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{i,m}; r_{m,k'}); \bar{x}^k \leftarrow \hat{p}_{k',k}; \hat{p}_{k,j} && \text{[by Lem. 4.4]}\\
&\quad= \textstyle\sum_{k'} \sum_k \mathsf{do}\ \bar{x}^k \leftarrow (\mathsf{do}\ \bar{x}^{k'} \leftarrow (\mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{i,m}; r_{m,k'}); \hat{p}_{k',k}); \hat{p}_{k,j} && \text{[by Lem. 4.1]}\\
&\quad= \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow \Big(\sum_{k'} \mathsf{do}\ \bar{y}^m \leftarrow q^{m-1}_{i,m}; \bar{x}^{k'} \leftarrow r_{m,k'}; \hat{p}_{k',k}\Big); \hat{p}_{k,j} && \text{[by Lem. 4.1]}\\
&\quad\geqslant \textstyle\sum_k \mathsf{do}\ \bar{x}^k \leftarrow (\mathsf{init}\, \bar{y}^m \leftarrow q^{m-1}_{i,m}\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^m \to r_{m,k}); \hat{p}_{k,j} && \text{[by 4.15]}\\
&\quad= \textstyle\sum_k \mathsf{init}\, \bar{y}^m \leftarrow q^{m-1}_{i,m}\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^m \to (\mathsf{do}\ \bar{x}^k \leftarrow r_{m,k}; \hat{p}_{k,j}) && \text{[by Lem. 4.1]}\\
&\quad\geqslant \mathsf{init}\, \bar{y}^m \leftarrow q^{m-1}_{i,m}\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^m \to (\mathsf{do}\ \bar{y}^l \leftarrow q_{m,l}; r_{l,j}) && \text{[by ind.]}\\
&\quad= \mathsf{do}\ \bar{y}^l \leftarrow (\mathsf{init}\, \bar{y}^m \leftarrow q_{i,m}\ \mathsf{in}(q^{m-1}_{m,m})^\star\ \mathsf{res}\, \bar{y}^l \to q^{m-1}_{m,l}); r_{l,j} && \text{[by Lem. 4.1]}
\end{aligned}$$

We have thus completed the proof of (4.13), and therefore we are done. □

Lemma 4.6. *Let p be a pattern-matching program; let $\bar{r} = \langle r_1, \dots, r_n\rangle$ be a finite sequence of pattern-matching programs; let $(n, \lambda i.\, \bar{y}^i, \lambda i.\, \lambda j.\, q_{i,j})$ be a computational net whose transitive* *closure is* $(n, \lambda i.\, \bar{y}^i, \lambda i.\, \lambda j.\, \hat{q}_{i,j})$, *and the following condition holds for all* i*:*

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{x} \leftarrow r_i; p = \sum\nolimits_j \mathsf{do}\ \bar{y}^j \leftarrow q_{i,j}; r_j. \tag{4.16}$$

If, moreover, for every i, j, $\mathrm{Vars}(p) \cap \mathrm{Vars}(\bar{y}^i) \subseteq \mathrm{Vars}(\bar{x})$ *and* $\mathrm{Vars}(r_i) \cap \mathrm{Vars}(\bar{y}^j) \subseteq \mathrm{Vars}(\bar{y}^i)$ *then for every* i*,*

$$\vdash_{\mathsf{ME}^\star} \mathsf{init}\,\bar{x} \leftarrow r_i\,\mathsf{in}\,p^\star = \sum\nolimits_j \mathsf{do}\ \bar{y}^j \leftarrow \hat{q}_{i,j}; r_j \tag{4.17}$$

Proof. Let $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$ be the computational net where for every i, $\bar{x}^i \doteq \bar{x}$ and for every i, j, $p_{i,j} \doteq p$. Let for every i, j, $r_{i,j} := r_i$. Then (4.16) can be rewritten to

$$\vdash_{\mathsf{ME}^\star} \sum\nolimits_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; p_{k,j} = \sum\nolimits_k \mathsf{do}\ \bar{y}^k \leftarrow q_{i,k}; r_{k,j}. \tag{4.18}$$

Let $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, \hat{p}_{i,j})$ be the transitive closure of $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$. By Lemma 4.5, for every i, j,

$$\vdash_{\mathsf{ME}^\star} \sum\nolimits_k \mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} = \sum\nolimits_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{i,k}; r_{k,j}. \tag{4.19}$$

Recall that $r_{i,k} \doteq r_i$, $\bar{x}^k \doteq \bar{x}$. It can easily be seen that for every i, j, $\hat{p}_{i,j}$ is provably equal to $(\mathsf{init}\,\bar{x} \leftarrow \mathsf{ret}\,\bar{x}\,\mathsf{in}\,p^\star)$. Therefore for every i, j, k, $\vdash_{\mathsf{ME}^\star}$ $\mathsf{do}\ \bar{x}^k \leftarrow r_{i,k}; \hat{p}_{k,j} = \mathsf{init}\,\bar{x} \leftarrow r_i\,\mathsf{in}\,p^\star$ and thus (4.19) is equivalent to the goal (4.17). □

Lemma 4.7. *Let* q *be a pattern-matching program; let* $\bar{r} = \langle r_1, \dots, r_n \rangle$ *be a finite sequence of pattern-matching programs; and let* $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, p_{i,j})$ *be a computational net with the transitive closure* $(n, \lambda i.\, \bar{x}^i, \lambda i.\, \lambda j.\, \hat{p}_{i,j})$ *such that the following condition holds for all* i*:*

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{y} \leftarrow q; r_i = \sum\nolimits_j \mathsf{do}\ \bar{x}^j \leftarrow r_j; p_{j,i}. \tag{4.20}$$

If, moreover, for every i, j, $\mathrm{Vars}(p_{i,j}) \cap \mathrm{Vars}(\bar{y}) \subseteq \mathrm{Vars}(\bar{x}^i)$ *then for every* i*,*

$$\vdash_{\mathsf{ME}^\star} \mathsf{init}\,\bar{y} \leftarrow \mathsf{ret}\,\bar{y}\,\mathsf{in}\,q^\star\,\mathsf{res}\,\bar{y} \to r_i = \sum\nolimits_j \mathsf{do}\ \bar{x}^j \leftarrow r_j; \hat{p}_{j,i}. \tag{4.21}$$

Proof. Let $(n, \lambda i.\, \bar{y}^i, \lambda i.\, \lambda j.\, q_{i,j})$ be the computational net where for every i, j, $\bar{y}^i \doteq \bar{y}$ and $q_{i,j} \doteq q$. Let for all i, j, $r_{i,j} \doteq r_j$. Then (4.20) can be rewritten to

$$\vdash_{\mathsf{ME}^\star} \sum\nolimits_k \mathsf{do}\ \bar{y}^k \leftarrow q_{j,k}; r_{k,i} = \sum\nolimits_k \mathsf{do}\ \bar{x}^k \leftarrow r_{j,k}; p_{k,i}. \tag{4.22}$$

Let $(n, \lambda i.\, \bar{y}^i, \lambda i.\, \lambda j.\, \hat{q}_{i,j})$ be the transitive closure of $(n, \lambda i.\, \bar{y}^i, \lambda i.\, \lambda j.\, q_{i,j})$. By Lemma 4.5, for every i, j,

$$\vdash_{\mathsf{ME}^\star} \sum\nolimits_k \mathsf{do}\ \bar{y}^k \leftarrow \hat{q}_{j,k}; r_{k,i} = \sum\nolimits_k \mathsf{do}\ \bar{x}^k \leftarrow r_{j,k}; \hat{p}_{k,i}. \tag{4.23}$$

Recall that $r_{k,i} \doteq r_i$, $\bar{y}^k \doteq \bar{y}$. It can easily be seen that for every j, k, $\hat{q}_{j,k}$ is provably equal to $(\mathsf{init}\,\bar{y} \leftarrow \mathsf{ret}\,\bar{y}\,\mathsf{in}\,q^\star)$, and thus (4.23) is equivalent to the goal (4.21). □

After instantiating n with 1 in the latter lemma, we obtain

Corollary 4.8. *Let for some appropriately typed programs p, q, r:*

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{x} \leftarrow r; p = \mathrm{do}\ \bar{y} \leftarrow q; r \tag{4.24}$$

and $\mathrm{Vars}(\bar{y}) \cap \mathrm{Vars}(r) \subseteq \mathrm{Vars}(\bar{x})$*. Then:*

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\, \bar{x} \leftarrow r\, \mathrm{in}\, p^\star = \mathrm{init}\, \bar{y} \leftarrow \mathrm{ret}\, \bar{y}\, \mathrm{in}\, q^\star\, \mathrm{res}\, \bar{y} \to r. \tag{4.25}$$

The proof of the Commutation Lemma is reminiscent of the celebrated Kleene's theorem about the equivalence of finite automata and regular expressions (cf. e.g. [Koz97]). In fact, we can make this point more precise by showing how the difficult direction of this theorem can be obtained as a consequence of Lemma 4.6.

Suppose we are given a nondeterministic finite state machine (FSM) A with n states, and a transition function $\sigma : S^2 \to \mathcal{P}(\Xi)$ where $S := \{1, \dots, n\}$ is the set of states and Ξ is the input alphabet. Let b be the initial state of A. For simplicity we assume that there is only one final state e. Now let us interpret Ξ as the underlying signature of the metalanguage of effects by attaching to every $a \in \Xi$ the type $1 \to T1$. Let us instantiate the data from the statement of Lemma 4.6 as follows:

– for every i let $r_i := \mathrm{ret}\langle\varnothing, \dots, \mathrm{ret}\,\star, \dots, \varnothing\rangle$ where the only component distinct from $\varnothing$ is on the i-th position;

– let $p := \sum_{i,j} \sum_{a \in \sigma(i,j)} \mathrm{do}\ x_i; a; r_j$;

– let for every i, j, $q_{i,j} := \sum_{a \in \sigma(i,j)} a$;

– all the vectors $\bar{y}^i$ collapse into one variable of the unit type, and hence we omit them in bindings.

It is easy to see that the commutativity condition (4.16) is fulfilled. Therefore, by Lemma 4.8 we have the presentation (4.17). After inserting both sides of (4.17) in the context $(\mathrm{do}\ \bar{x} \leftarrow \square; x_e)$, instantiating i by b and performing evident simplification, we obtain:

$$\mathrm{init}\, \bar{x} \leftarrow r_b\, \mathrm{in} \left(\sum_{i,j} \sum_{a \in \sigma(i,j)} \mathrm{do}\ x_i; a; r_j\right)^\star \mathrm{res}\, \bar{x} \to x_e = \hat{q}_{b,e}.$$

We refer to the left and the right sides of this equality by M and L correspondingly. The equation $M = L$ can now be read as follows. The left-hand side M essentially uses the facilities of the metalanguage of effects. Specifically, the ret operation cannot be eliminated from M easily, as it is essential for encoding the original FSM A as a ME expression. On the other hand, L does not contain non-trivial occurrences of ret operator, e.g. L can be seen as a regular expression in the classical sense. The fact that the equality $M = L$ is provable in $\mathsf{ME}^\star$ means that the regular expression L we have constructed generates just the same language as the one recognised by the original

FSM A. This can be justified as follows. It is easy to see that $\mathsf{NF}(M)$ is the language recognised by A and $\mathsf{NF}(L)$ is the language generated by L. By Corollary 3.35 we have the equivalence $\mathsf{NF}(M) \simeq_\star \mathsf{NF}(L)$, which in this case by definition amounts to the equality of sets $\mathsf{NF}(M) = \mathsf{NF}(L)$.

4.2 Reduction to Kleene algebra

The goal of this section is to prove completeness of provable equality in $\mathsf{ME}^\star$ over a continuous Kleene monad in case the programs come from a rather restricted class which we call *tight programs*. The idea of the proof is to relate programs with regular expressions and then make use of a completeness theorem for equality of regular expressions over the algebra of regular events. The reduction mechanism will also easily imply decidability of program equality.

The discussion in this section involves some terms and facts concerning Kleene algebra, regular expressions, etc. Let us briefly recall the necessary information (it can also be found in more detail in [Koz94]).

Let Ξ be an arbitrary set of symbols. Then the set of regular expressions over Ξ is defined by the BNF:

$$E ::= a \mid \varnothing \mid 1 \mid E \cdot E \mid E + E \mid E^*$$

where a ranges over Ξ. The *algebra of regular events* over Ξ is the set of all sets of finite strings over Ξ (including the empty one, denoted by λ) under the usual set-theoretic operations. We can interpret any regular expression over the corresponding algebra of regular events by operator ρ defined by the assignments:

- $\rho(a) := \{a\}$ if $a \in \Xi$;
- $\rho(1) := \{\lambda\}$;
- $\rho(\varnothing) := \emptyset$;
- $\rho(a + b) := \rho(a) \cup \rho(b)$;
- $\rho(a \cdot b) := \{st \mid s \in \rho(a), t \in \rho(b)\}$;
- $\rho(a^\star) := \{s \ldots s \mid s \in \rho(a)\}$.

The algebra of regular events is an instance of the general notion of *Kleene algebra*, a structure subject to the axioms presented in Fig. 4.1. An important result about the calculus of Kleene algebra is the following theorem, proved by Kozen.

Theorem 4.9. [Koz94] *The equational theory of Kleene algebras is complete over the algebra of regular events.*

We proceed now with the main text. Before we formally introduce the class of programs for which we have announced the completeness result, we shall prove several facts about generic programs. Recall that we have previously called ret-free the programs not containing ret operator. The following definition generalises this notion slightly.

$$
\begin{array}{cccc}
a + \varnothing = a & a + b = b + a & a + a = a & a + (b + c) = (a + b) + c \\
\varnothing \cdot a = \varnothing & (a + b) \cdot c = a \cdot c + b \cdot c & a \cdot \varnothing = \varnothing & (a + b) \cdot c = a \cdot c + b \cdot c
\end{array}
$$

$$
a \cdot b \leqslant b \implies a^* \cdot b \leqslant b \qquad\qquad a \cdot b \leqslant a \implies a \cdot b^* \leqslant a
$$

FIGURE 4.1: Kleene algebra axioms.

Definition 4.10 (Almost ret-freeness)**.** We call programs of the form $\mathrm{ret}\, p$ with cartesian p *administrative*. A program is *almost* ret*-free* if every subterm $\mathrm{ret}\, r$ of it is administrative.

Almost ret-free programs are obviously closed under reductions by $\rightarrowtail_{\mathsf{km}}$. The notions and notation of Section 3.4 become more simple in the case of almost ret-free programs. In particular, both the relations $\simeq_\star$ and $\simeq_\omega$ over almost ret-free shallow-deterministic programs coincide with the usual α-equivalence. Operator prg, defined on page 86, can be described in the ret-free case as follows. Given an almost ret-free program p of a computational return type, let $\mathsf{nf}_\sigma(p) \doteq \sum_i p_i$ where the p_i do not contain the choice operator on top. Then, by definition, $\mathsf{prg}(p)$ returns precisely the set of those p_i which are deterministic.

Given a program p with a computational return type, let $\mathsf{tr}(p) := \bigcup \{\mathsf{prg}(q) \mid p \rightarrowtail_{\mathsf{k}}^\star q\}$. Elementary properties of tr include:

- $\mathsf{tr}(\varnothing) = \emptyset$;
- $\mathsf{tr}(p + q) = \mathsf{tr}(p) \cup \mathsf{tr}(q)$;
- $\mathsf{tr}(p) = \{p\}$ if p is deterministic;
- $\mathsf{tr}(\mathrm{do}\ x \leftarrow p; q) = \{\mathrm{do}\ x \leftarrow s; t \mid s \in \mathsf{tr}(p), t \in \mathsf{tr}(q)\}$;
- $\mathsf{tr}(\mathrm{init}\, x \leftarrow p\, \mathrm{in}\, q^\star) = \bigcup_i \mathsf{tr}(\mathrm{init}\, x \leftarrow p\, \mathrm{in}\, q^i)$.

Operator tr is similar to NF but it does not call normalisation under $\rightarrowtail_{\mathsf{m}}$. The relationship between tr and NF can be justified by the following:

Lemma 4.11. *Let p be an almost* ret*-free program and let r be any program. Then*

$$
\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r) \simeq_\star \{\mathsf{nf}(\mathrm{do}\ x \leftarrow t; \mathrm{ret}\, r) \mid t \in \mathsf{tr}(p)\}. \tag{4.26}
$$

Proof. Let us show first that

$$
\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r) \simeq_\star \bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r) \mid p \rightarrowtail_{\mathsf{k}}^\star q\}. \tag{4.27}
$$

First observe that by definition of NF and by (3.21), for every t,

$$\mathsf{NF}(t) = \bigcup \{\gamma(s) \mid t \rightarrowtail^{\star}_{\mathsf{km}} s\} \gtrsim_{\star} \{\gamma(s) \mid t \rightarrowtail^{\star}_{\mathsf{k}} s\} \gtrsim_{\star} \bigcup_i \gamma(\mathsf{unf}_i(t)) \simeq_{\star} \mathsf{NF}(t).$$

Therefore, $\mathsf{NF}(t) \simeq_{\star} \bigcup \{\gamma(s) \mid t \rightarrowtail^{\star}_{\mathsf{k}} s\}$. After instantiating t with $(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r)$, by definition of $\rightarrowtail_{\mathsf{k}}$, we obtain

$$\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r) \simeq_{\star} \bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, u) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q, r \rightarrowtail^{\star}_{\mathsf{k}} u\}.$$

Hence (4.27) can be equivalently replaced with

$$\bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, u) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q, r \rightarrowtail^{\star}_{\mathsf{k}} u\} \simeq_{\star} \bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q\}.$$

Clearly, the left-hand side is greater than the right-hand side. In order to show the converse, we fix some q and u such that $p \rightarrowtail^{\star}_{\mathsf{k}} q, r \rightarrowtail^{\star}_{\mathsf{k}} u$ and some $t \in \gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, u)$. By Remark 3.23, $\mathsf{nf}(q)$ must be of the form $\sum_i q_i$. Since by definition, $\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, u) = \bigcup_i \gamma(\mathrm{do}\ x \leftarrow q_i; \mathrm{ret}\, u)$, there is i such that $t \in \gamma(\mathrm{do}\ x \leftarrow q_i; \mathrm{ret}\, u)$. By Lemma 3.26, q_i must be deterministic, i.e. either $q_i \doteq (\mathrm{do}\ \bar{x} \leftarrow \bar{a}; a)$ or $q_i \doteq (\mathrm{do}\ \bar{x} \leftarrow \bar{a}; \mathrm{ret}\, w)$ where a and all the a_i are atomic. W.l.o.g. it suffices to consider only the latter case because the former one can be reduced to it by switching to $(\mathrm{do}\ \bar{x} \leftarrow \bar{a}; z \leftarrow a; \mathrm{ret}\, z)$. Therefore we have $t \in \gamma(\mathrm{do}\ \bar{x} \leftarrow \bar{a}; \mathrm{ret}\, u[w/x])$. It follows by induction from Lemma 3.28 that $t \in \gamma(\mathrm{do}\ \bar{x} \leftarrow \bar{a}; \mathrm{ret}\, r[w/x]) = \gamma(\mathrm{do}\ x \leftarrow q_i; r) \lesssim_{\star} \bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q\}$. The proof of (4.27) is thus completed.

Another fact needed to prove the claim is the equivalence

$$\mathsf{prg}(\mathsf{nf}(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r)) \simeq_{\star} \{\mathsf{nf}(t) \mid t \in \mathsf{prg}(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r)\} \tag{4.28}$$

which can be easily proved for every almost ret-free q by induction over term complexity of q. Now the proof of (4.26) runs as follows:

$$\begin{aligned}
&\mathsf{NF}(\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, r) \\
&\quad \simeq_{\star} \bigcup \{\gamma(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q\} && [\text{by } 4.27] \\
&\quad \simeq_{\star} \bigcup \{\mathsf{prg}(\mathsf{nf}(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r)) \mid p \rightarrowtail^{\star}_{\mathsf{k}} q\} && [\text{by def. of } \gamma] \\
&\quad \simeq_{\star} \bigcup \{\mathsf{nf}(t) \mid t \in \mathsf{prg}(\mathrm{do}\ x \leftarrow q; \mathrm{ret}\, r), p \rightarrowtail^{\star}_{\mathsf{k}} q\} && [\text{by } 4.28] \\
&\quad \simeq_{\star} \bigcup \{\mathsf{nf}(\mathrm{do}\ x \leftarrow t; \mathrm{ret}\, r) \mid t \in \mathsf{prg}(q), p \rightarrowtail^{\star}_{\mathsf{k}} q\} && [\text{by def. of } \mathsf{prg}] \\
&\quad \simeq_{\star} \{\mathsf{nf}(\mathrm{do}\ x \leftarrow t; \mathrm{ret}\, r) \mid t \in \mathsf{tr}(p)\}. && [\text{by def. of } \mathsf{tr}]
\end{aligned}$$

The proof is completed. □

Recall that for almost ret-free programs $\simeq_{\star}$ turns into α-equivalence. Then by taking $r := x$ in Lemma 4.11 we obtain by (3.20) and definition of nf

Corollary 4.12. *For every almost* ret*-free program p whose return type is computational,* $\mathsf{NF}(p) = \{\mathsf{nf}(t) \mid t \in \mathsf{tr}(p)\}$.

Definition 4.13 (Flatness)**.** Given an atomic program a and a vector of distinct variables $\bar{x}$ we denote by $a_{i \triangleright \bar{x}}$ the program $(\text{do } x_i \leftarrow a; \text{ret}\, \bar{x})$ if $0 < i \leqslant |\bar{x}|$ and $(\text{do } a; \text{ret}\, \bar{x})$ if $i = 0$. For every vector of distinct variables $\bar{x}$ we define the class of *$\bar{x}$-flat* programs by induction as follows.

FL1. Every $a_{i \triangleright \bar{x}}$, $\text{ret}\, \bar{x}$ and $\varnothing$ are $\bar{x}$-flat;

FL2. $(p + q)$ is $\bar{x}$-flat, provided both p and q are $\bar{x}$-flat;

FL3. $(\text{do } \bar{y} \leftarrow p; q)$ is $\bar{x}$-flat, provided p is $\bar{y}$-flat and q is $\bar{x}$-flat.

FL4. $(\text{init}\, \bar{x} \leftarrow p \,\text{in}\, q^\star)$ is $\bar{x}$-flat, provided both p and q are $\bar{x}$-flat.

We call a program simply *flat* if it is $\bar{x}$-flat for some $\bar{x}$. If some flat program p happens to be $\bar{x}$-flat with some $\bar{x}$, we refer to $\bar{x}$ as a *footprint* of p.

The class of flat programs is by definition smaller than the class of almost ret-free programs. Modulo the first unit law it is also wider than the class of ret-free programs: we can switch from a ret-free program to a flat program by replacing every atomic $a : TA$ with $a_{1 \triangleright v_A}$ where $v_A : A$ is a fresh variable name uniquely corresponding to the type A.

Definition 4.14 (Tightness)**.** We call a flat deterministic program p *tight* if all atomic subterms of p are normal and for every subterm u of p having form $(\text{do } \bar{x} \leftarrow s; t)$, $\text{Vars}(t) = \text{Vars}(\bar{x})$ unless s is an atomic program (i.e. $u \doteq a_{i \triangleright \bar{x}}$ for some atomic a). We call any flat program p tight if all the (deterministic) programs from $\mathsf{tr}(p)$ are tight.

Immediate properties of tight programs include the following.

- If $u \doteq (\text{do } \bar{x} \leftarrow p; q)$ is tight, then $\text{Vars}(u) = \text{Vars}(p)$.
- If $u \doteq (\text{init}\, \bar{x} \leftarrow p \,\text{in}\, q^\star)$ is tight, then $\text{Vars}(u) = \text{Vars}(p)$.
- If $(p + q)$ is flat, then it is tight iff both p, q are tight.
- If $(\text{do } \bar{x} \leftarrow p; q)$ is flat, then it is tight iff p, q are tight and $\text{Vars}(q) = \text{Vars}(\bar{x})$.
- If $(\text{init}\, \bar{x} \leftarrow p \,\text{in}\, q^\star)$ is flat, then it is tight iff p, q are tight and $\text{Vars}(q) = \text{Vars}(\bar{x})$.

The requirement for atomic programs of a tight program to be normal ensures the following important property which we will commonly use henceforth.

Lemma 4.15. *Given a deterministic tight program p,* $\text{Vars}(p) = \text{Vars}(\mathsf{nf}(p))$.

Proof. It can easily be seen that rewriting under the associativity law neither spoils tightness nor affects the set of free variables. Therefore, w.l.o.g. p is of the form

$$\mathsf{do}\ \bar{x}^1 \leftarrow w_1; \ldots ; \bar{x}^n \leftarrow w_n; w$$

where every w_i is either $a^i_{k_i \triangleright \bar{x}^i}$ or $\mathsf{ret}\, \bar{x}^i$ and w is either $a_{k \triangleright \bar{x}}$ or $\mathsf{ret}\, \bar{x}$. Let us cancel from p all the program sections of the form $\bar{x}^i \leftarrow \mathsf{ret}\, \bar{x}^i$, replace every program section of the form $\bar{x}^i \leftarrow a^i_{k_i \triangleright \bar{x}^i}$ by $x^i_{k_i} \leftarrow a^i$ and denote the result by q. Obviously, $p \rightarrowtail^\star_\beta q$ and $\mathrm{Vars}(p) = \mathrm{Vars}(q)$. Observe that q must be of the form $(\mathsf{do}\ \bar{z} \leftarrow \bar{b}; u)$ where all the b_i are atomic and u is either atomic or equal to $\mathsf{ret}\, \bar{x}$. In the former case q is normal, which means $q = \mathsf{nf}(p)$, and we are done. The latter case follows in the same way unless $q \doteq (\mathsf{do}\ z_1 \leftarrow b_1; \ldots ; z_m \leftarrow b_m; \mathsf{ret}\, z_m)$ or $q \doteq (\mathsf{do}\ z_1 \leftarrow b_1; \ldots ; z_m \leftarrow b_m; \mathsf{ret}\, e_E)$ and z_m is of type $E \in \mathcal{U}$. Both these forms are only one step behind the normal form, which is $(\mathsf{do}\ z_1 \leftarrow b_1; \ldots ; z_{m-1} \leftarrow b_{m-1}; b_m)$ and of course $\mathrm{Vars}(q) = \mathrm{Vars}(\mathsf{nf}(q))$. Therefore, $\mathrm{Vars}(p) = \mathrm{Vars}(\mathsf{nf}(p))$, and we are done. □

The idea behind the definition of tightness is that we capture thereby the class of those programs which can be associated with Kleene algebra terms. We would like to make this association precise and then prove the completeness and decidability result by reduction to Theorem 4.9. In order to do so, we will need to introduce one more condition. Informally speaking, we need to make sure that the bound variable names used in programs obey some definite global convention. This will allow us to ignore α-equivalence and stick with the syntactic equality of terms.

Definition 4.16 (Name pool). A *name pool* is a set of fresh variable names Λ equipped with a well-ordering $\preceq$ isomorphic to ω and such that Λ contains infinitely many variables of any type from Type^T_W.

Let us denote by $\min_A \Lambda$ the smallest element of Λ under $\preceq$, whose type is A. Let us also define by induction: $\min_A \Lambda := \min^0_A \Lambda$ and $\min^{n+1}_A \Lambda := \min^n_A (\Lambda \backslash \min_A \Lambda)$. For every $\kappa \in \Lambda$ let us denote by Λ^κ the initial segment $\{v \preceq \kappa\}$ of Λ.

Let Λ be a name pool. Given a $\bar{z}$-flat tight deterministic program p and an injective substitution $\sigma : \mathrm{Vars}(p) \to \Lambda$, we recursively define a flat program $\mathsf{nf}^\sigma_\Lambda(p)$, α-equivalent to $p\sigma$ by the following assignments respecting the applicable clauses of Definition 4.13. Expression $\sigma \triangledown \varsigma$ here denotes overlapping of σ by ς, i.e. $\sigma \triangledown \varsigma$ sends every x to $x\sigma$ if $x\varsigma = x$ and to $x\varsigma$ otherwise.

FL1. $\mathsf{nf}^\sigma_\Lambda(\mathsf{ret}\, \bar{z}) := \mathsf{ret}\, \bar{z}\sigma$,

$\mathsf{nf}^\sigma_\Lambda(a_{0 \triangleright \bar{z}}) := (a\sigma)_{0 \triangleright \bar{z}\sigma}$,

$\mathsf{nf}^\sigma_\Lambda(a_{i \triangleright \bar{z}}) := (a\sigma)_{i \triangleright \bar{z}\varsigma}$ where $i > 0$, $\varsigma := \sigma \triangledown [v/z_i]$, $v := \min_A(\Lambda \backslash \mathrm{Vars}(\bar{z}_{\hat{\imath}}\sigma))$ and A is the type of z_i.

FL3. $\mathsf{nf}^{\sigma}_{\Lambda}(\mathrm{do}\ \bar{x} \leftarrow p; q) := \mathrm{do}\ \bar{v} \leftarrow \mathsf{nf}^{\sigma}_{\Lambda}(p); \mathsf{nf}^{\varsigma}_{\Lambda}(q)$ where $\mathsf{nf}^{\sigma}_{\Lambda}(p)$ is $\bar{v}$-flat and $\varsigma := [\bar{v}/\bar{x}]$.

Definition 4.17 (Λ-normality)**.** Given a name pool Λ, we call a tight deterministic program Λ-*normal* if it belongs to the codomain of $\mathsf{nf}^{\sigma}_{\Lambda}$ for some appropriate σ, q. More generally, we call a tight program p Λ-*normal* if every element of $\mathsf{tr}(p)$ is Λ-normal.

Lemma 4.18. *Let $w \doteq (\mathrm{do}\ \bar{v} \leftarrow p; q)$ be a tight program such that both p and q are Λ-normal and p is not atomic (i.e. w is not of the form $a_{i \triangleright \bar{z}}$). Then w is itself Λ-normal.*

Proof. First consider the deterministic case. By definition, there exist programs u, r and injective substitutions $\sigma : \mathrm{Vars}(u) \to \Lambda$, $\theta : \mathrm{Vars}(r) \to \Lambda$ such that $p \doteq \mathsf{nf}^{\sigma}_{\Lambda}(u)$ and $q \doteq \mathsf{nf}^{\theta}_{\Lambda}(r)$. Let $\bar{x}$ be the footprint of u. Since w is tight, $\mathrm{Vars}(r\theta) = \mathrm{Vars}(q) = \mathrm{Vars}(\bar{v})$. Let s be the program obtained from r by replacing every variable $x \in \mathrm{Vars}(r)$ with $(x\theta)[\bar{x}/\bar{v}]$ throughout. Then $\mathrm{Vars}(s) = \mathrm{Vars}(\bar{x})$. We prove that $w \doteq \mathsf{nf}^{\sigma}_{\Lambda}(\mathrm{do}\ \bar{x} \leftarrow u; s)$. Indeed, by definition, $\mathsf{nf}^{\sigma}_{\Lambda}(\mathrm{do}\ \bar{x} \leftarrow u; s) \doteq (\mathrm{do}\ \bar{v} \leftarrow \mathsf{nf}^{\sigma}_{\Lambda}(u); \mathsf{nf}^{\varsigma}_{\Lambda}(s))$ where $\varsigma = [\bar{v}/\bar{x}]$. Observe that s is α-equivalent to $(r\theta)[\bar{x}/\bar{v}]$. Therefore, by definition of $\mathsf{nf}^{\varsigma}_{\Lambda}$ and $\mathsf{nf}^{\theta}_{\Lambda}$, $\mathsf{nf}^{\varsigma}_{\Lambda}(s) \doteq (r\theta)[\bar{x}/\bar{v}][\bar{v}/\bar{x}] \doteq r\theta \doteq \mathsf{nf}^{\theta}_{\Lambda}(r)$ and we are done by the calculation:

$$\mathsf{nf}^{\sigma}_{\Lambda}(\mathrm{do}\ \bar{x} \leftarrow u; s) \doteq (\mathrm{do}\ \bar{v} \leftarrow \mathsf{nf}^{\sigma}_{\Lambda}(u); \mathsf{nf}^{\theta}_{\Lambda}(r)) \doteq (\mathrm{do}\ \bar{v} \leftarrow p; q).$$

Now consider the general case. Every element of $\mathsf{tr}(w)$ has the form $(\mathrm{do}\ \bar{v} \leftarrow s; t)$ with $s \in \mathsf{tr}(p)$, $t \in \mathsf{tr}(q)$ and is inherently tight. The programs s and t are both deterministic Λ-normal. As we have argued above, $(\mathrm{do}\ \bar{v} \leftarrow s; t)$ must be Λ-normal. By definition of Λ-normality, w is Λ-normal. □

Lemma 4.19. *Let $s \doteq (\mathrm{do}\ \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow p; q); r)$ be a tight Λ-normal program with respect to some name pool Λ, and p, q and r are not atomic. Then the program $t \doteq (\mathrm{do}\ \bar{y} \leftarrow p; \bar{x} \leftarrow q; r)$ is also tight and Λ-normal. Moreover, $\vdash_{\mathsf{ME}^+} s = t$.*

Proof. By the definition of flatness, p is $\bar{y}$-flat and q is $\bar{x}$-flat. Obviously this implies flatness of t. Let us prove that t is tight. Since p, q, r are inherently tight we only need to show that $\mathrm{Vars}(r) = \mathrm{Vars}(\bar{x})$ and $\mathrm{Vars}(\mathrm{do}\ \bar{x} \leftarrow q; r) = \mathrm{Vars}(\bar{y})$. The former equation directly follows from the tightness of s. The latter one holds because, again, by the tightness of s, $\mathrm{Vars}(\mathrm{do}\ \bar{x} \leftarrow q; r) = \mathrm{Vars}(q)$ and $\mathrm{Vars}(\bar{y}) = \mathrm{Vars}(q)$.

Let us show how Λ-normality of s implies Λ-normality of t. First consider the case when p, q and r are all deterministic. Then s must be the image under $\mathsf{nf}^{\sigma}_{\Lambda}$ of some program $(\mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u; w); l)$ for some injective substitution σ, which by definition implies: $p \doteq \mathsf{nf}^{\sigma}_{\Lambda}(u)$, $q \doteq \mathsf{nf}^{\varsigma}_{\Lambda}(w)$, $r \doteq \mathsf{nf}^{\theta}_{\Lambda}(l)$ where $\varsigma := [\bar{v}/\bar{y}]$, $\theta := [\bar{z}/\bar{x}]$. As a result p, q and r are Λ-normal. By Lemma 4.18, t is Λ-normal. In general, if s is not deterministic, we are done by the same argument applied to all the components of $\mathsf{tr}(s)$.

Finally, observe that $\vdash_{\mathsf{ME}^+} s = t$ follows from Lemma 4.1, whose conditions are satisfied since by tightness $\mathrm{Vars}(r) = \mathrm{Vars}(\bar{x})$. □

Lemma 4.20. *Given a name pool Λ, let p and q be two deterministic Λ-normal flat programs for which $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$. Then the footprints of p and q coincide.*

Proof. Let $\bar{x}$ be the footprint of p and let $\bar{y}$ be the footprint of q. We transform p provably equivalently under ME^{+} as follows. First let us reduce p by Lemma 4.19 to the form $(\mathrm{do}\ \bar{x}^1 \leftarrow w_1; \ldots; \bar{x}^n \leftarrow w_n; w)$ where every w_i is either $a_{k \triangleright \bar{x}^i}$ for some atomic a or $\mathrm{ret}\,\bar{x}^i$, and w is either $a_{k \triangleright \bar{x}}$ for some atomic a or $\mathrm{ret}\,\bar{x}$. Then we cancel from the result all the program sections of the form $\bar{x}^i \leftarrow \mathrm{ret}\,\bar{x}^i$. Finally, if $w \doteq a_{k \triangleright \bar{x}}$ then we replace it by $(\mathrm{do}\ \bar{x} \leftarrow a_{k \triangleright \bar{x}}; \mathrm{ret}\,\bar{x})$ and p takes the form $(\mathrm{do}\ \bar{x}^1 \leftarrow a^1_{k_1 \triangleright \bar{x}^1}; \ldots; \bar{x}^n \leftarrow a^n_{k_n \triangleright \bar{x}^n}; \mathrm{ret}\,\bar{x}^n)$ where all the a^i are atomic. Note that Λ-normality was maintained under these transformations. Analogously we reduce q to the form $(\mathrm{do}\ \bar{y}^1 \leftarrow b^1_{l_1 \triangleright \bar{y}^1}; \ldots; \bar{y}^m \leftarrow b^m_{l_m \triangleright \bar{y}^m}; \mathrm{ret}\,\bar{y}^m)$. It is straightforward that $n = m$. Let us show by induction over n that $\bar{x}^n = \bar{y}^n$. This will complete the proof. If $n = 0$ then we are trivially done.

Consider the case when $n > 0$. Let $p' := (\mathrm{do}\ \bar{x}^2 \leftarrow a^2_{k_2 \triangleright \bar{x}^2}; \ldots; \bar{x}^n \leftarrow a^n_{k_n \triangleright \bar{x}^n}; \mathrm{ret}\,\bar{x}^n)$ and $q' := (\mathrm{do}\ \bar{y}^2 \leftarrow b^2_{l_2 \triangleright \bar{y}^2}; \ldots; \bar{y}^n \leftarrow b^n_{l_n \triangleright \bar{y}^n}; \mathrm{ret}\,\bar{y}_n)$. If $k_1 = l_1 = 0$ then the assumption $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$ implies $\mathsf{nf}(p') \equiv \mathsf{nf}(q')$ and we are done by induction hypothesis.

Let us show that the situation when precisely one of k_1, l_1 equals 0 is impossible. Let e.g. $k_1 = 0, l_1 \neq 0$. The assumption $\mathsf{nf}(p) \equiv \mathsf{nf}(q)$ implies $\mathsf{nf}(p') \equiv \mathsf{nf}(q'[v/y^1_{l_1}])$ where v is some fresh variable, and we have:

$$\begin{aligned}&\mathrm{Vars}(\bar{x}^1) = \mathrm{Vars}(p') = \mathrm{Vars}(\mathsf{nf}(p')) = \\ &\mathrm{Vars}(\mathsf{nf}(q'[v/y^1_{l_1}])) = \mathrm{Vars}(q'[v/y^1_{l_1}]) = \mathrm{Vars}(\bar{y}^1[v/y^1_{l_1}]).\end{aligned}$$

Therefore, $v \in \mathrm{Vars}(\bar{x})$ which contradicts the choice of v.

Finally, assume that both k_1 and l_1 are positive. Then $\mathsf{nf}(p'[v/x^1_{k_1}]) \equiv \mathsf{nf}(q'[v/y^1_{l_1}])$ for some fresh variable v which in conjunction with the tightness of p and q implies:

$$\begin{aligned}&\mathrm{Vars}(\bar{x}^1[v/x^1_{k_1}]) = \mathrm{Vars}(p'[v/x^1_{k_1}]) = \mathrm{Vars}(\mathsf{nf}(p'[v/x^1_{k_1}])) = \\ &\mathrm{Vars}(\mathsf{nf}(q'[v/y^1_{l_1}])) = \mathrm{Vars}(q'[v/y^1_{l_1}]) = \mathrm{Vars}(\bar{y}^1[v/y^1_{l_1}])\end{aligned}$$

and therefore: $\mathrm{Vars}(\bar{x}^1_{\hat{k}_1}) = \mathrm{Vars}(\bar{y}^1_{\hat{l}_1})$. Since, by assumption, both p and q are Λ-normal, $x^1_{k_1} = \min_C(\Lambda \backslash \mathrm{Vars}(\bar{x}^1_{\hat{k}_1})) = \min_C(\Lambda \backslash \mathrm{Vars}(\bar{y}^1_{\hat{l}_1})) = y^1_{l_1}$ where C is the return type of $x^1_{k_1}$ and $y^1_{l_1}$. Therefore $\mathsf{nf}(p') \equiv \mathsf{nf}(q')$ and thus we are done by induction hypothesis. $\square$

Remark 4.21. Many proofs about flat programs can be slightly facilitated if the program under consideration satisfies the condition: in every subterm $(\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^{\star})$ of it $p \doteq \mathrm{ret}\,\bar{x}$. In fact we can always ensure this condition by transforming the original program under the reduction rule

$$\mathrm{init}\,\bar{x} \leftarrow p\,\mathrm{in}\,q^{\star} \rightarrowtail p + \mathrm{do}\ \bar{x} \leftarrow p; (\mathrm{init}\,\bar{x} \leftarrow \mathrm{ret}\,\bar{x}\,\mathrm{in}\,q^{\star})$$

applied once to every redex. Evidently this transformation is sound under $\mathsf{ME}^\star$, and it spoils neither flatness nor tightness. Actually, it will not spoil many other properties of flat programs that we are going to introduce.

Lemma 4.22. *Let Λ be a name pool. Then for every $\bar z$-flat tight program p there exists an effectively computable $\kappa(p) \in \Lambda$ such that for every $\kappa \geq \kappa(p)$ and every pair of injective substitutions $\sigma : \mathrm{Vars}(p) \to \Lambda^\kappa$, $\varsigma : \mathrm{Vars}(\bar z) \to \Lambda^\kappa$ there is an effectively computable program $p_{\sigma,\varsigma}$ possessing the following properties:*

1. *every $p_{\sigma,\varsigma}$ is tight $\bar z\varsigma$-flat;*
2. *every $p_{\sigma,\varsigma}$ is Λ-normal;*
3. *$\vdash_{\mathsf{ME}^\star} p\sigma = \sum_\varsigma p_{\sigma,\varsigma}$;*
4. *if for all $t \in \mathrm{tr}(p\sigma)$, $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma)$ then for all $t \in \mathrm{tr}(p_{\sigma,\varsigma})$, $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma)$.*

Proof. By Remark 4.21 we transform the original programs p so that Kleene star occurs in it only in the form $(\mathrm{init}\,\bar x \leftarrow \mathrm{ret}\,\bar x\,\mathrm{in}\,q^\star)$. It is clear that the reduction specified in Remark 4.21 does respect the properties (1)–(4). Then we define the element $\kappa(p) \in \Lambda$ in question by induction over the term complexity of p in respect of the clauses of Definition 4.13.

FL1. If $p \doteq \varnothing$ then we define $\kappa(p) := \min \Lambda$, $p_{\sigma,\varsigma} := \varnothing$ for every σ, ς and the claim becomes trivial.

For $p \doteq \mathrm{ret}\,\bar z$ we put $\kappa(p) := \min \Lambda$. The injections σ, ς become instantiated to $\sigma, \varsigma : \mathrm{Vars}(\bar z) \to \Lambda^\kappa$. Let us put $p_{\sigma,\varsigma} := p\sigma$ if $\sigma = \varsigma$ and $p_{\sigma,\varsigma} := \varnothing$ otherwise. The conditions (1)–(4) are easily verified.

Similarly, if $p \doteq a_{0 \triangleright \bar z}$ then $\kappa(p) := \min \Lambda$ and for every pair of injections σ, ς : $\mathrm{Vars}(\bar z) \to \Lambda^\kappa$ we put $p_{\sigma,\varsigma} := (a\sigma)_{0 \triangleright \bar z\sigma}$ if $\sigma = \varsigma$ and $p_{\sigma,\varsigma} := \varnothing$ otherwise.

If $p \doteq a_{i \triangleright \bar z}$ and $i > 0$ then $\kappa(p) := \min_A^{|\bar z|} \Lambda$ where A is the type z_i. The injections σ, ς take form $\sigma : \mathrm{Vars}(a) \cup \mathrm{Vars}(\bar z_{\hat\imath}) \to \Lambda^\kappa$ and $\varsigma : \mathrm{Vars}(a) \cup \mathrm{Vars}(\bar z) \to \Lambda^\kappa$. Let us define $p_{\sigma,\varsigma} := (a\sigma)_{i \triangleright \bar z\varsigma}$ if $\varsigma = \sigma \triangledown [v/z_i]$, $v = \min_A(\Lambda \backslash \mathrm{Vars}(\bar z_{\hat\imath}\sigma))$ and $p_{\sigma,\varsigma} := \varnothing$ otherwise. As we can see, every $p_{\sigma,\varsigma}$ is either $\mathrm{nf}^\sigma_\Lambda(p)$ or $\varnothing$. This observation immediately implies the conditions (1), (2), and (4). In order to establish (3), we need to prove that for every σ at least one of the $p_{\sigma,\varsigma}$ is distinct from $\varnothing$. Equivalently, we need to prove that $\min_A(\Lambda \backslash \mathrm{Vars}(\bar z_{\hat\imath}\sigma)) \in \Lambda^\kappa$. The latter follows from the calculation: $\min_A(\Lambda \backslash \mathrm{Vars}(\bar z_{\hat\imath}\sigma)) \preceq \min_A^{|\bar z|} \Lambda = \kappa(p) \preceq \kappa$.

FL2. Let $p \doteq (q + r)$. By induction hypothesis there exist $\kappa(q)$ and $\kappa(r)$, satisfying the statement of the lemma for q and r. Then we put $\kappa(p) := \max\{\kappa(q), \kappa(r)\}$. If $\mathrm{tr}(q) = \emptyset$ then we put $p_{\sigma,\varsigma} := r_{\sigma_2,\varsigma}$ and if $\mathrm{tr}(r) = \emptyset$ then we put $p_{\sigma,\varsigma} := q_{\sigma_1,\varsigma}$. In both these cases, (1)–(4) follow trivially by induction. We assume henceforth that both $\mathsf{tr}(q)$ and $\mathsf{tr}(r)$ are nonempty.

Every substitution $\sigma : \mathrm{Vars}(p) \to \Lambda^\kappa$ being restricted to $\mathrm{Vars}(q)$ and $\mathrm{Vars}(r)$ correspondingly generates substitutions $\sigma_1 : \mathrm{Vars}(q) \to \Lambda^\kappa$ and $\sigma_2 : \mathrm{Vars}(r) \to \Lambda^\kappa$. Since by definition $\kappa \geq \kappa(q), \kappa \geq \kappa(r)$, by induction hypothesis, for every $\varsigma : \mathrm{Vars}(\bar{z}) \to \Lambda^\kappa$ there exists appropriate $q_{\sigma_1,\varsigma}$ and $r_{\sigma_2,\varsigma}$. We define $p_{\sigma,\varsigma} := q_{\sigma_1,\varsigma} + r_{\sigma_2,\varsigma}$.

The conditions (1)–(3) can be verified easily. Let us prove (4). Suppose that for every $t \in \mathsf{tr}(p\sigma)$, $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma)$, let $s \in \mathsf{tr}(p_{\sigma,\varsigma})$ and prove that $\mathrm{Vars}(s) = \mathrm{Vars}(p\sigma)$. Since $\mathsf{tr}(p_{\sigma,\varsigma}) = \mathsf{tr}(q_{\sigma_1,\varsigma}) \cup \mathsf{tr}(r_{\sigma_2,\varsigma})$, s should come either from $\mathsf{tr}(q_{\sigma_1,\varsigma})$ or from $\mathsf{tr}(r_{\sigma_2,\varsigma})$. Due to symmetry it suffices to prove only the former case.

Every t from $\mathsf{tr}(q\sigma_1)$ must belong to $\mathsf{tr}(p\sigma)$. Therefore, according to the assumption, for every $t \in \mathsf{tr}(q\sigma_1)$, $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma) \supseteq \mathrm{Vars}(q\sigma_1)$. On the other hand, for every $t \in \mathsf{tr}(q\sigma_1)$, $\mathrm{Vars}(t) \subseteq \mathrm{Vars}(q\sigma_1)$. We have thus proved that

$$\forall t \in \mathrm{Vars}(q\sigma_1).\, \mathrm{Vars}(t) = \mathrm{Vars}(q\sigma_1).$$

By applying the induction hypothesis we obtain: $\mathrm{Vars}(s) = \mathrm{Vars}(q\sigma_1)$. We will be done once we show that $\mathrm{Vars}(q\sigma_1) = \mathrm{Vars}(p\sigma)$. Since $\mathrm{Vars}(p\sigma) = \mathrm{Vars}(q\sigma_1) \cup \mathrm{Vars}(r\sigma_2)$ it suffices to establish the inclusion $\mathrm{Vars}(p\sigma) \subseteq \mathrm{Vars}(q\sigma_1)$. By assumption, $\mathsf{tr}(q)$ is nonempty. Therefore $\mathsf{tr}(q\sigma_1)$ is also nonempty, i.e. there is at least one $t \in \mathsf{tr}(q\sigma_1) \subseteq \mathsf{tr}(p\sigma)$. By the assumption $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma)$. On the other hand, $\mathrm{Vars}(t) \subseteq \mathrm{Vars}(q\sigma_1)$ and we can conclude the inclusion in question.

FL3. Let $p \doteq (\mathsf{do}\ \bar{x} \leftarrow q; r)$. Let us again put $\kappa(p) := \max\{\kappa(q), \kappa(r)\}$. If $q \doteq \varnothing$ then $p_{\sigma,\varsigma} := \varnothing$ and (1)–(4) are trivially satisfied. We assume henceforth that q is distinct from $\varnothing$. Since p is tight, $\mathrm{Vars}(p) = \mathrm{Vars}(q)$. Therefore the substitution σ takes the form: $\sigma : \mathrm{Vars}(q) \to \Lambda^\kappa$. Since $\kappa \geq \kappa(p) \geq \kappa(q)$, for every injection $\theta : \mathrm{Vars}(\bar{x}) \to \Lambda^\kappa$ there exists $q_{\sigma,\theta}$ as required by the induction invariant. According to the induction hypothesis there exist correctly defined $r_{\theta,\varsigma}$. We put therewith

$$p_{\sigma,\varsigma} := \textstyle\sum_\theta \mathsf{do}\ \bar{x}^\theta \leftarrow q_{\sigma,\theta}; r_{\theta,\varsigma} \tag{4.29}$$

where $\bar{x}^\theta := \bar{x}\theta$. Let us prove the conditions (1)–(4).

1. By induction, every $r_{\theta,\varsigma}$ is $\bar{z}\varsigma$-flat. Therefore, by the definition of flatness, $p_{\sigma,\varsigma}$ is also $\bar{z}\varsigma$-flat. We prove tightness by showing that every element of the sum in (4.29) is tight. Since r is tight, for every $t \in \mathsf{tr}(r)$, $\mathrm{Vars}(t) = \mathrm{Vars}(r)$. Therefore, for every $t \in \mathsf{tr}(r\theta)$, $\mathrm{Vars}(t) = \mathrm{Vars}(r\theta)$. By induction hypothesis, for every $t \in \mathsf{tr}(r_{\theta,\varsigma})$, $\mathrm{Vars}(t) = \mathrm{Vars}(r\theta)$. Since p is tight, $\mathrm{Vars}(r\theta) = \mathrm{Vars}(\bar{x}^\theta)$. Therefore $(\mathsf{do}\ \bar{x}^\theta \leftarrow q_{\sigma,\theta}; r_{\theta,\varsigma})$ is tight, and we are done.

2. By induction hypothesis and Lemma 4.18, every $(\mathsf{do}\ \bar{x}^{\theta} \leftarrow q_{\sigma,\theta}; r_{\theta,\varsigma})$ must be Λ-normal. Observe that every $t \in \mathsf{tr}(p_{\sigma,\varsigma})$ belongs to $\mathsf{tr}(\mathsf{do}\ \bar{x}^{\theta} \leftarrow q_{\sigma,\theta}; r_{\theta,\varsigma})$ for some θ. Therefore, by the definition of Λ-normality every $p_{\sigma,\varsigma}$ is Λ-normal.
3. The proof is done by the calculation:

$$\begin{aligned} p\sigma &\doteq \mathsf{do}\ \bar{x} \leftarrow q\sigma; r \\ &= \textstyle\sum_{\theta} \mathsf{do}\ \bar{x} \leftarrow q_{\sigma,\theta}; r && \text{[by ind.]} \\ &= \textstyle\sum_{\theta} \mathsf{do}\ \bar{x}^{\theta} \leftarrow q_{\sigma,\theta}; r[\bar{x}^{\theta}/\bar{x}] \\ &\doteq \textstyle\sum_{\theta} \mathsf{do}\ \bar{x}^{\theta} \leftarrow q_{\sigma,\theta}; r\theta \\ &= \textstyle\sum_{\varsigma}\sum_{\theta} \mathsf{do}\ \bar{x}^{\theta} \leftarrow q_{\sigma,\theta}; r_{\theta,\varsigma} && \text{[by ind.]} \\ &= \textstyle\sum_{\varsigma} p_{\sigma,\varsigma} && \text{[by 4.29]} \end{aligned}$$

4. Let for all $t \in \mathsf{tr}(p\sigma)$, $\mathrm{Vars}(t) = \mathrm{Vars}(p\sigma)$ and let $s \in \mathsf{tr}(p_{\sigma,\varsigma})$. We need to show that $\mathrm{Vars}(s) = \mathrm{Vars}(p\sigma)$. By the definition of tr, s must be of the form $(\mathsf{do}\ \bar{x}^{\theta} \leftarrow t; w)$ where $t \in \mathsf{tr}(q_{\sigma,\theta})$, $w \in \mathsf{tr}(r_{\theta,\varsigma})$. By induction hypothesis, $\mathrm{Vars}(t) = \mathrm{Vars}(q\sigma)$. Using the fact that both s and p are tight, the equality in question is now proven as follows:

$$\mathrm{Vars}(s) = \mathrm{Vars}(t) = \mathrm{Vars}(q\sigma) = \mathrm{Vars}(p\sigma).$$

FL4. Suppose $p \doteq (\mathsf{init}\, \bar{z} \leftarrow \mathsf{ret}\, \bar{z}\, \mathsf{in}\, q^{\star})$. Let $\kappa(p) := \kappa(q)$. Let us enumerate all the injective substitutions from $\mathrm{Vars}(\bar{z}) \to \Lambda^{\kappa}$ and denote the i-th one by σ_i. Recall that by tightness, $\mathrm{Vars}(q) = \mathrm{Vars}(\bar{z})$. Therefore, both σ and ς must be among the σ_i. Let $\bar{v}^i := \bar{z}\sigma_i$ and let $q_{i,j} := q_{\sigma_i,\sigma_j}$. By induction hypothesis, for every i:

$$\vdash_{\mathsf{ME}^{\star}} \mathsf{do}\ \bar{z} \leftarrow \mathsf{ret}\, \bar{z}\sigma_i; q = \textstyle\sum_j \mathsf{do}\ \bar{v}^j \leftarrow q_{i,j}; \mathsf{ret}\, \bar{z}\sigma_j$$

and for every i, j, k, $\mathrm{Vars}(\bar{v}^i) \cap \mathrm{Vars}(q_{j,k}) \subseteq \mathrm{Vars}(\bar{v}^j)$, $\mathrm{Vars}(\bar{v}^i) \cap \mathrm{Vars}(\mathsf{ret}\, z\sigma_j) \subseteq \mathrm{Vars}(\bar{v}^j)$, $\mathrm{Vars}(q) \cap \mathrm{Vars}(\bar{v}^i) \subseteq \mathrm{Vars}(\bar{z})$. By Lemma 4.6 and by **(unit$_1$)**, **(unit$_2$)** there exist programs $\hat{q}_{i,j}$ such that:

$$\vdash_{\mathsf{ME}^{\star}} p\sigma_i = \mathsf{do}\ \bar{z} \leftarrow \mathsf{ret}\, \bar{z}\sigma_i; q = \textstyle\sum_j \mathsf{do}\ \bar{v}^j \leftarrow \hat{q}_{i,j}; \mathsf{ret}\, \bar{z}\sigma_j = \sum_j \hat{q}_{i,j}.$$

Let us put for every i, j, $p_{\sigma_i,\sigma_j} := \hat{q}_{i,j}$. It is immediately clear that for every i, j, p_{σ_i,σ_j} is $\bar{v}^j$-flat, and the condition (3) is satisfied. In order to establish (1) and (2), we prove by further induction over k that every $q^k_{i,j}$ is Λ-normal, and for every $t \in \mathsf{tr}(q_{i,j})$, $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i)$.

Induction Base. By definition, $q^0_{i,j} \doteq \mathsf{ret}\, \bar{v}^i + q_{i,i}$ if $i = j$ and $q^0_{i,j} \doteq q_{i,j}$ if $i \neq j$. In both cases Λ-normality follows by the outer induction. Let $t \in \mathsf{tr}(q^0_{i,i})$, i.e. $t \in \mathsf{tr}(q_{i,i}) \cup \{\mathsf{ret}\, \bar{v}^i\}$. If $t \doteq \mathsf{ret}\, \bar{v}^i$ then of course $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i)$. Otherwise,

$t \in \mathrm{Vars}(q^0_{i,i})$ and $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i)$ by tightness of p. Similarly, if $t \in \mathsf{tr}(q^0_{i,j})$ and $i \neq j$, $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i)$.

Induction Step. According to the definition,

$$q^k_{i,j} \doteq q^{k-1}_{i,j} + \mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^\star\,\mathrm{res}\,\bar{v}^k \to q^{k-1}_{k,j}.$$

Let $t \in \mathsf{tr}(q^k_{i,j}) = \mathsf{tr}(q^{k-1}_{i,j}) \cup \bigcup_m \mathsf{tr}\big(\mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^m\,\mathrm{res}\,\bar{v}^k \to q^{k-1}_{k,j}\big)$. We need to prove that t is Λ-normal and $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i)$. If t falls into $\mathsf{tr}(q^{k-1}_{i,j})$ we are immediately done by the outer induction. Consider the remaining case: $t \in \mathsf{tr}\big(\mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^m\,\mathrm{res}\,\bar{v}^k \to q^{k-1}_{k,j}\big)$ for some m. It is straightforward to show by further induction over m that $\big(\mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^m\big)$ is tight, and for every $s \in \mathsf{tr}\big(\mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^m\big)$, $\mathrm{Vars}(s) = \mathrm{Vars}(\bar{v}^k)$. Now, by definition, $\big(\mathrm{init}\,\bar{v}^k \leftarrow q^{k-1}_{i,k}\,\mathrm{in}(q^{k-1}_{k,k})^m\,\mathrm{res}\,\bar{v}^k \to q^{k-1}_{k,j}\big)$ also must be tight. By the successive application of Lemma 4.18, we can show that it is also Λ-normal. Therefore t is also Λ-normal. Finally, $\mathrm{Vars}(t) = \mathrm{Vars}(q^{k-1}_{i,k}) = \mathrm{Vars}(\bar{v}^k)$.

We have at the moment ensured (1), (2) and (3). In order to show (4), recall that we have proved that for every $t \in \mathsf{tr}(\hat{q}_{i,j}) = \mathsf{tr}(p_{\sigma_i,\sigma_j})$, $\mathrm{Vars}(t) = \mathrm{Vars}(\bar{v}^i) = \mathrm{Vars}(\bar{z}\sigma_i) = \mathrm{Vars}(p\sigma_i)$. This proves the righ-hand side of the implication (4), and thus it proves the whole implication. □

Lemma 4.23. *Given tight programs $p, q : TA$, the problems $\vdash_{\mathsf{ME}^\omega} p = q$ and $\vdash_{\mathsf{ME}^\star} p = q$ are equivalent and decidable.*

Proof. Let Λ be some name pool. First we argue that it suffices to proceed with the assumption that p and q are both Λ-normal and have the same footprint. For an arbitrarily chosen substitution $\sigma : \mathrm{Vars}(p) \cup \mathrm{Vars}(q) \to \Lambda$ we can equivalently switch from p, q to $p\sigma, q\sigma$. By Lemma 4.22, $p\sigma$ and $q\sigma$ can be effectively and equivalently under $\mathsf{ME}^\star$ reduced to the forms $\sum_i p_i$ and $\sum_i q_i$ respectively, where for every i, p_i and q_i are Λ-normal and have the same footprint uniquely defined by i. Then

$$\begin{aligned}
&\vdash_{\mathsf{ME}^\omega} p = q \\
&\quad\iff \textstyle\bigcup_i \mathsf{NF}(p_i) = \bigcup_i \mathsf{NF}(q_i) && \text{[by Corollary 3.35]} \\
&\quad\iff \forall i.\ \forall s \in \mathsf{tr}(p_i).\ \exists j.\ \exists t \in \mathsf{tr}(q_j).\ \mathsf{nf}(s) \equiv \mathsf{nf}(t)\ \wedge \\
&\qquad\quad\ \forall i.\ \forall s \in \mathsf{tr}(q_i).\ \exists j.\ \exists t \in \mathsf{tr}(p_j).\ \mathsf{nf}(s) \equiv \mathsf{nf}(t) && \text{[by Corollary 4.12]} \\
&\quad\iff \forall i.\ \forall s \in \mathsf{tr}(p_i).\ \exists t \in \mathsf{tr}(q_i).\ \mathsf{nf}(s) \equiv \mathsf{nf}(t)\ \wedge \\
&\qquad\quad\ \forall i.\ \forall s \in \mathsf{tr}(q_i).\ \exists t \in \mathsf{tr}(p_i).\ \mathsf{nf}(s) \equiv \mathsf{nf}(t) && \text{[by Lemma 4.20]} \\
&\quad\iff \forall i. \mathsf{NF}(p_i) = \mathsf{NF}(q_i) && \text{[by Corollary 4.12]} \\
&\quad\iff \forall i.\ \vdash_{\mathsf{ME}^\star} p_i = q_i && \text{[by Corollary 3.35]}
\end{aligned}$$

Therefore, if we proved the equivalence $\vdash_{\mathsf{ME}^\star} p_i = q_i \iff \vdash_{\mathsf{ME}^\omega} p_i = q_i$ for every i, the claim would follow from the circular chain of implications:

$$\begin{aligned} \vdash_{\mathsf{ME}^\omega} p = q \implies \forall i.\ \vdash_{\mathsf{ME}^\omega} p_i = q_i \implies \forall i.\ \vdash_{\mathsf{ME}^\star} p_i = q_i \\ \implies \vdash_{\mathsf{ME}^\star} p = q \implies \vdash_{\mathsf{ME}^\omega} p = q. \end{aligned}$$

We continue henceforth under the assumption that both p and q are tight $\bar{z}$-flat.

By definition, $\vdash_{\mathsf{ME}^\star} p = q$ implies $\vdash_{\mathsf{ME}^\omega} p = q$. Assume $\vdash_{\mathsf{ME}^\omega} p = q$ and prove $\vdash_{\mathsf{ME}^\star} p = q$. By the definition of Λ-normality, both $\mathsf{tr}(p)$ and $\mathsf{tr}(q)$ consist of Λ-normal programs. In the remainder we shall be guided by the diagram:

$$\begin{tikzcd} \vdash_{\mathsf{ME}^\omega} p = q \arrow[d, "f"] & \vdash_{\mathsf{ME}^\star} h(f(p)) = h(f(q)) \\ \rho(f(p)) = \rho(f(q)) \arrow[r, dotted] & \vdash f(p) = f(q) \arrow[u, "h"'] \end{tikzcd}$$

Here, f and h respectively denote translations of program equalities to equalities of regular expressions and vice versa. The operator ρ is the standard interpretation of regular expressions over the algebra of regular events. The dotted line on the bottom of the diagram refers to Kozen's completeness proof (i.e. Theorem 4.9). All the arrows except the bottom one shall be defined below (recall that TA is the return type of p and q).

For any normal atomic program a we introduce a new symbol $\hat{a}$, and for every pair (a, x) where a is a normal atomic program with some return type C and a variable $x \in \Lambda$ of type C, we introduce the new symbols $\hat{a}_x$. Let f be the function, recursively sending tight programs to terms of Kleene algebra over the signature

$$\Xi := \{\hat{a}_x \mid a \in \mathcal{A}_\Sigma\} \cup \{\hat{a} \mid a \in \mathcal{A}_\Sigma\}$$

according to the assignments:

- $f(\varnothing) := \varnothing$;
- $f(\mathrm{ret}\,\bar{x}) := 1$;
- $f(a_{i \triangleright \bar{x}}) := \hat{a}_{x_i}$ $(i > 0)$;
- $f(a_{0 \triangleright \bar{x}}) := \hat{a}$;
- $f(s + t) := f(s) + f(t)$;
- $f(\mathrm{do}\ \bar{x} \leftarrow s; t) := f(s) \cdot f(t)$;
- $f(\mathrm{init}\,\bar{x} \leftarrow s\,\mathrm{in}\,t^\star) := f(s) \cdot (f(t))^\star$.

Prove that the sets $\rho(f(p))$ and $\rho(f(q))$ are equal, i.e. the equation $f(p) = f(q)$ holds over the algebra of regular events. Since $\vdash_{\mathsf{ME}^\omega} p = q$, by Corollary 3.35, $\mathsf{NF}(p)$ and $\mathsf{NF}(q)$ must be pointwise α-equivalent. By Corollary 4.12 $\mathsf{NF}(p) = \{\mathsf{nf}(s) \mid s \in \mathsf{tr}(p)\}$ and $\mathsf{NF}(q) = \{\mathsf{nf}(t) \mid t \in \mathsf{tr}(q)\}$. Therefore, for every $s \in \mathsf{tr}(p)$ there exists $t \in \mathsf{tr}(q)$ such that $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ and vice versa. Show that for any such pair (s,t):

$$\rho(f(s)) = \rho(f(t)). \tag{4.30}$$

More generally, we prove (4.30) for every pair (s,t) of deterministic tight Λ-normal programs such that $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$. Let us proceed by induction over m, the common number of maximal atomic subterms (i.e. those atomic subterms which are not proper subterms of any other atomic subterms) of s and t. If $m = 0$ then by the definition of f and ρ, $\rho(f(s)) = \{\lambda\} = \rho(f(t))$. Suppose $m > 0$. Let us reduce s and t to one of the forms: $\mathsf{ret}\,\bar{z}$, $a_{i\triangleright\bar{z}}$ or $(\mathsf{do}\ \bar{x} \leftarrow a_{i\triangleright\bar{x}}; r)$ so that the number of maximal atomic subterms does not increase, making sure that the premise of induction remains valid. To that end, consider s and suppose it is not yet in the necessary form. Then there are two options:

1. $s \doteq (\mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{x}; r)$. By the definition of Λ-normality, r is Λ normal. Also, it can easily be seen that $\rho(f(s)) = \rho(f(r))$. Therefore we can switch from s to r and the premise of induction does hold for the pair (r,t).

2. $s \doteq (\mathsf{do}\ \bar{x} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow r; u); w)$ with r being non-atomic. Then we switch from s to $s' := (\mathsf{do}\ \bar{y} \leftarrow r; \bar{x} \leftarrow u; w)$. Let us show that the premise of induction remains valid for the pair (s',t). Obviously $\rho(f(s)) = \rho(f(s'))$. By Lemma 4.19 s' is Λ-normal, tight and $\vdash_{\mathsf{ME}^+} s' = s$. Therefore, by Theorem 2.17 $\mathsf{nf}(s') \equiv \mathsf{nf}(s) \equiv \mathsf{nf}(t)$.

Successively applying the reductions (1)–(2), we eventually reach one of the specified forms for s. In the same manner we reduce r. Termination of these reductions is guaranteed by the fact that the relation $\rightarrowtail_\beta$, which is implicit here, is strongly normalising.

Let us now proceed with the induction step. Consider the possible cases.

1. $s \doteq t \doteq \mathsf{ret}\,\bar{z}$. Trivially, $\rho(f(s)) = \rho(f(t))$.

2. $s \doteq a_{i\triangleright\bar{z}}$ and $t \doteq b_{j\triangleright\bar{z}}$. The assumption $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ immediately implies $a \equiv b$, $i = j$ and the equation (4.30) becomes trivial.

3. $s \doteq a_{i\triangleright\bar{z}}$ and $t \doteq (\mathsf{do}\ \bar{x} \leftarrow b_{j\triangleright\bar{x}}; r)$. The assumption $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ implies $a \equiv b$. Obviously r may not contain signature symbols, because otherwise the identity $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ would fail. It can easily be seen by induction hypothesis that $\mathsf{nf}(r) \doteq \mathsf{ret}\,z$ and $\rho(f(r)) = \{\lambda\}$. The identity $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ takes the form $\mathsf{nf}(a_{i\triangleright\bar{z}}) \equiv \mathsf{nf}(a_{j\triangleright\bar{z}})$. By Λ-normality, $i = j$. If both i,j are 0, then $\rho(f(s)) = \{\hat{a}\} = \rho(\hat{a}\cdot\lambda) = \rho(f(t))$. If both i,j are distinct from 0, then $\rho(f(s)) = \{\hat{a}_{i\triangleright\bar{z}}\} = \rho(\hat{a}_{i\triangleright\bar{z}}\cdot\lambda) = \rho(f(t))$. The proof of (4.30) is thus complete.

4. $s \doteq (\mathsf{do}\ \bar{x} \leftarrow a_{i\triangleright\bar{x}}; r)$ and $t \doteq b_{j\triangleright\bar{z}}$. This case is treated symmetrically to the previous one.

5. $s \doteq (\mathsf{do}\ \bar{x} \leftarrow a_{i\triangleright\bar{x}}; r)$ and $t \doteq (\mathsf{do}\ \bar{y} \leftarrow b_{j\triangleright\bar{y}}; u)$. Obviously, $a \doteq b$. Let us assume that one of i, j, e.g. i is equal to 0 and prove that the other one is too. By contradiction: let $j \neq 0$. Then $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ implies $\mathsf{nf}(\mathsf{do}\ a; r) \equiv \mathsf{nf}(\mathsf{do}\ y_j \leftarrow a; u)$ and thus $\mathsf{nf}(r) \equiv \mathsf{nf}(u[v/y_j])$ where v is some fresh variable. Note that $\mathrm{Vars}(\bar{x}) = \mathrm{Vars}(r) = \mathrm{Vars}(\mathsf{nf}(r)) = \mathrm{Vars}(\mathsf{nf}(u[v/y_j])) = \mathrm{Vars}(u[v/y_j]) = \mathrm{Vars}(\bar{y}[v/y_j])$ and thus $v \in \mathrm{Vars}(\bar{x})$. But v was a brand new variable. Contradiction. We have thus proved that once either i or j is 0 then the other one is too. Suppose this is the case. Then $\rho(f(s)) = \{\hat{a} \cdot w \mid w \in \rho(f(r))\}$, $\rho(f(t)) = \{\hat{a} \cdot w \mid w \in \rho(f(u))\}$ and we are done by induction hypothesis.

   Suppose neither i nor j is equal to 0. Then the assumption $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$ takes the form $\mathsf{nf}(\mathsf{do}\ x_i \leftarrow a; r) \equiv \mathsf{nf}(\mathsf{do}\ y_j \leftarrow a; u)$. Therefore $\mathsf{nf}(r[v/x_i]) \equiv \mathsf{nf}(u[v/y_j])$ where v is a fresh variable. Then $\mathrm{Vars}(\bar{x}[v/x_i]) = \mathrm{Vars}(r[v/x_i]) = \mathrm{Vars}(\mathsf{nf}(r[v/x_i])) = \mathrm{Vars}(\mathsf{nf}(u[v/y_j])) = \mathrm{Vars}(\bar{y}[v/y_j])$ and therefore $\mathrm{Vars}(\bar{x}_{\hat{\imath}}) = \mathrm{Vars}(\bar{y}_{\hat{\jmath}})$. Since both s and t are Λ-normal, $x_i = \min_C(\Lambda \backslash \mathrm{Vars}(\bar{x}_{\hat{\imath}})) = \min_C(\Lambda \backslash \mathrm{Vars}(\bar{y}_{\hat{\jmath}})) = y_j$ where C is the return type of x_i. As a consequence: $\rho(f(s)) = \{\hat{a}_{x_i} \cdot w \mid w \in \rho(f(r))\}$, $\rho(f(t)) = \{\hat{a}_{x_i} \cdot w \mid w \in \rho(f(u))\}$ and $\mathsf{nf}(r) \equiv \mathsf{nf}(u)$. By definition of Λ-normality both r and u are Λ-normal. We are done by calling the induction invariant to (r, u).

It can easily be seen that the remaining options for s and t (e.g. $s \doteq \mathsf{ret}\,\bar{z}$, $t \doteq a_{i\triangleright\bar{z}}$) do not satisfy the condition $\mathsf{nf}(s) \equiv \mathsf{nf}(t)$. Therefore, the proof of (4.30) is completed.

It can easily be seen by induction that for every flat u,

$$\rho(f(u)) = \bigcup\nolimits_{t \in \mathsf{tr}(u)} \rho(f(t)). \tag{4.31}$$

Now the identity $\rho(f(p)) = \rho(f(q))$ is shown as follows:

$$\begin{aligned} \rho(f(p)) &= \bigcup\nolimits_{s \in \mathsf{tr}(p)} \rho(f(s)) && \text{[by 4.31]} \\ &= \bigcup\nolimits_{t \in \mathsf{tr}(q)} \rho(f(t)) && \text{[by 4.30]} \\ &= \rho(f(q)). && \text{[by 4.31]} \end{aligned}$$

By Theorem 4.9 the equality $f(p) = f(q)$ must be provable in the Kleene algebra calculus. Let $\bar{v}$ be the vector of all bound variables occurring in p, q, and for every i, let A_i be the type of v_i. We define a function h recursively translating any Kleene algebra term t over Ξ to a flat program $v_1 : A_1, \ldots, v_n : A_n \triangleright h(t) : T(A_1 \times \ldots \times A_n)$ as follows:

- $h(\varnothing) := \varnothing$;
- $h(1) := \mathsf{ret}\,\bar{v}$;
- $h(\hat{a}) := a_{0\triangleright\bar{v}}$;
- $h(\hat{a}_x) := a_{i\triangleright\bar{v}}$ where $x = v_i$;
- $h(p + q) := h(p) + h(q)$;
- $h(p \cdot q) := \mathsf{do}\ \bar{v} \leftarrow h(p); h(q)$;
- $h(p^\star) := \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,h(p)^\star$.

It is easy to see that h, when applied to the axioms and derivation rules of Kleene algebra, produces theorems and admissible rules of the calculus $\mathsf{ME}^\star$, which means that $h(f(p)) = h(f(q))$ is provable in $\mathsf{ME}^\star$.

Now we are left to show that for every $\bar{x}$-flat subterm t of either p or q,

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow h(f(t)); \mathrm{ret}\,\bar{x} = t$$

This will complete the proof of the implication: $\vdash_{\mathsf{ME}^\omega} p = q \implies \vdash_{\mathsf{ME}^\star} p = q$ as follows: $\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{v} \leftarrow h(f(p)); \mathrm{ret}\,\bar{z} = \mathrm{do}\ \bar{v} \leftarrow h(f(q)); \mathrm{ret}\,\bar{z} = q$. We proceed by induction over term complexity. By Definition 4.13, the following cases are possible:

FL1. Let $t \doteq a_{i \triangleright \bar{x}}$. Then $\vdash_{\mathsf{ME}^\star} (\mathrm{do}\ \bar{v} \leftarrow h(f(t)); \mathrm{ret}\,\bar{x}) = (\mathrm{do}\ \bar{v} \leftarrow a_{j \triangleright \bar{v}}; \mathrm{ret}\,\bar{x}) = a_{i \triangleright \bar{x}}$ where j is such that $x_i = v_j$ and 0 if $i = 0$. The cases $t \doteq \mathrm{ret}\,\bar{x}$ and $t \doteq \varnothing$ follow trivially.

FL2. Let $t \doteq (u + w)$. By Definition 4.13, both u and w must be $\bar{x}$-flat. Then $\vdash_{\mathsf{ME}^\star}$ $(\mathrm{do}\ \bar{v} \leftarrow h(f(t)); \mathrm{ret}\,\bar{x}) = (\mathrm{do}\ \bar{v} \leftarrow h(f(u)); \mathrm{ret}\,\bar{x}) + (\mathrm{do}\ \bar{v} \leftarrow h(f(w)); \mathrm{ret}\,\bar{x})$ and we are done by the induction hypothesis.

FL3. Let $t \doteq (\mathrm{do}\ \bar{y} \leftarrow u; w)$. By Definition 4.13, u is $\bar{y}$-flat and w is $\bar{x}$-flat. We are done by the following calculation in $\mathsf{ME}^\star$:

$$\begin{aligned}
&\mathrm{do}\ \bar{y} \leftarrow u; w\\
&\quad= \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow h(f(u)); \mathrm{ret}\,\bar{y}); w && \text{[by ind.]}\\
&\quad= \mathrm{do}\ \bar{v} \leftarrow h(f(u)); \bar{y} \leftarrow \mathrm{ret}\,\bar{y}; w && \text{[by Lemma 4.1]}\\
&\quad= \mathrm{do}\ \bar{v} \leftarrow h(f(u)); \bar{v} \leftarrow h(f(w)); \mathrm{ret}\,\bar{x} && \text{[by } \textbf{(unit}_1\textbf{)}\text{, ind.]}\\
&\quad= \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow h(f(u)); h(f(w))); \mathrm{ret}\,\bar{x} && \text{[by Lemma 4.1]}\\
&\quad= \mathrm{do}\ \bar{v} \leftarrow h(f(\mathrm{do}\ \bar{y} \leftarrow u; w)); \mathrm{ret}\,\bar{x} && \text{[by def. of } f, h\text{]}
\end{aligned}$$

FL4. Let $t \doteq (\mathrm{init}\,\bar{x} \leftarrow u\,\mathrm{in}\,w^\star)$. By Definition 4.13 both u and w are $\bar{x}$-flat. First note the following:

$$\begin{aligned}
&\mathrm{init}\,\bar{x}\ \leftarrow u\,\mathrm{in}\,w^\star\\
&\quad= \mathrm{init}\,\bar{x} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow h(f(u)); \mathrm{ret}\,\bar{x})\,\mathrm{in}\,w^\star && \text{[by ind.]}\\
&\quad= \mathrm{do}\ \bar{v} \leftarrow h(f(u)); \mathrm{init}\,\bar{x} \leftarrow \mathrm{ret}\,\bar{x}\,\mathrm{in}\,w^\star && \text{[by Lemma 4.1]}
\end{aligned}$$

Observe that $\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{x} \leftarrow \mathrm{ret}\,\bar{x}; w = \mathrm{do}\ \bar{v} \leftarrow h(f(w)); \mathrm{ret}\,\bar{x}$ and $\mathrm{Vars}(\mathrm{ret}\,\bar{x}) \cap \mathrm{Vars}(\bar{v}) \subseteq \mathrm{Vars}(\bar{x})$. Therefore by Corollary 4.8:

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\,\bar{x} \leftarrow \mathrm{ret}\,\bar{x}\,\mathrm{in}\,w^\star = \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,h(f(w))^\star\,\mathrm{res}\,\bar{v} \to \mathrm{ret}\,\bar{x} \tag{4.32}$$

and we can continue the previous calculation as follows:

$$\begin{aligned}
&\mathsf{do}\ \bar v \leftarrow h(f(u)); \mathsf{init}\,\bar x \leftarrow \mathsf{ret}\,\bar x\,\mathsf{in}\,w^\star \\
&\quad = \mathsf{do}\ \bar v \leftarrow h(f(u)); \mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}\,h(f(w))^\star \\
&\quad\ \mathsf{res}\,\bar v \to \mathsf{ret}\,\bar x && \text{[by 4.32]} \\
&\quad = \mathsf{do}\ \bar v \leftarrow (\mathsf{do}\ \bar v \leftarrow h(f(u)); \\
&\quad\ \mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}\,h(f(w))^\star); \mathsf{ret}\,\bar x && \text{[by Lem. 4.1]} \\
&\quad = \mathsf{do}\ \bar v \leftarrow h(f(\mathsf{init}\,\bar x \leftarrow u\,\mathsf{in}\,w^\star)); \mathsf{ret}\,\bar x && \text{[by def. of } f,h]
\end{aligned}$$

The last case is thus completed.

We have proved the equivalence $\vdash_{\mathsf{ME}^\omega} p = q \iff \vdash_{\mathsf{ME}^\star} p = q$. The decidability part is shown as follows. As we have seen $\vdash_{\mathsf{ME}^\omega} p = q$ implies validity of the equation $f(p) = f(q)$ over the algebra of regular events. On the other hand, if $f(p) = f(q)$ holds over the algebra of regular events, then by Theorem 4.9, $f(p) = f(q)$ must be provable in Kleene algebra calculus. As we have argued above, in this case $\vdash_{\mathsf{ME}^\star} p = q$ and therefore $\vdash_{\mathsf{ME}^\omega} p = q$. In summary, $p = q$ is provable in ME^ω iff both $f(p)$ and $f(q)$ generate the same regular language. The latter problem is decidable (cf. e.g. [RS97]). The proof is thus completed. □

4.3 Completeness and decidability for flat programs

In the previous section we have established a completeness and decidability result for a rather restricted class of programs. In order to make this result useful we need to extend it at least for flat programs. We achieve this by providing suitable reductions of flat programs to tight programs. These reductions are not straightforward and require some auxiliary notions together with some elementary results about them.

Definition 4.24 (Referential variable)**.** We say that a variable x is *referential* in a flat program p if for every subterm $a_{i \triangleright \bar x}$ of p, $x \neq x_i$.

Definition 4.25 (Cut-free programs)**.** We call a flat program *cut-free* if for every $\bar z$-flat subterm p of it every variable $z \notin \mathrm{Vars}(\bar z)$ is referential in p.

This idea behind the definition of cut-free programs can be explained as follows. Consider the program $(\mathsf{do}\ x \leftarrow (\mathsf{do}\ \langle x, y\rangle \leftarrow a_{2\triangleright\langle x,y\rangle}; \mathsf{ret}\,x); q)$. As we can see, q might depend on y but it is not going to be the same y to which the result of a was bound. That y was cut, and we cannot recover its value at the point of interest. The notion of being cut-free effectively rules out programs of this sort. E.g. for the example at hand, $y \notin \mathrm{Vars}(x)$ but y is not referential in $(\mathsf{do}\ \langle x, y\rangle \leftarrow a_{2\triangleright\langle x,y\rangle}; \mathsf{ret}\,x)$. It is should be intuitively clear from the definition that the cut-free programs are rather more well-behaved then the generic flat program. The exact benefit of using this notion will become apparent in the process of establishing the announced program reductions. Our plan is therefore as follows: first we shall show how to build a cut-free program by a flat program (Lemma 4.28); then we shall show how to reduce cut-free programs to tight programs (Lemma 4.29).

Lemma 4.26. *Let p be a $\bar{v}$-flat program, and let $\bar{v}$ be a vector of distinct variables such that* $\mathrm{Vars}(\bar{z}) \subseteq \mathrm{Vars}(\bar{v})$ *and all the variables from* $\mathrm{Vars}(\bar{v})\backslash\mathrm{Vars}(\bar{z})$ *are referential in p. Then*

$$\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v}.$$

Proof. We prove the claim by induction over term complexity of p with respect to the clauses of Definition 4.13.

FL1. If $p \doteq \varnothing$ or $p \doteq \mathrm{ret}\,\bar{v}$ then the claim becomes trivial. Let $p \doteq a_{i \triangleright \bar{v}}$ and $i > 0$. Note that by assumption of the lemma, $v_i \in \mathrm{Vars}(\bar{z})$. Therefore

$$\begin{aligned}
p &\doteq \mathrm{do}\ v_i \leftarrow a; \mathrm{ret}\,\bar{v} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ v_i \leftarrow a; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by Lemma 4.1]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ v_i \leftarrow a; \mathrm{ret}\,\bar{v}); \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by Lemma 4.1]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow a_{i \triangleright \bar{v}}; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} \\
&\doteq \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v}
\end{aligned}$$

The proof of the case $i = 0$ runs analogously.

FL2. Let $p \doteq (q + r)$. Then

$$\begin{aligned}
p &\doteq q + r \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} + \\
&\quad\ \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by ind.]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow (q + r); \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by } \mathbf{(dist_1^+)}\text{]} \\
&\doteq \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v}
\end{aligned}$$

and we are done.

FL3. Suppose $p \doteq (\mathrm{do}\ \bar{x} \leftarrow q; r)$. Let $\bar{y}$ be a vector of distinct variables such that $\mathrm{Vars}(\bar{y}) = \mathrm{Vars}(\bar{x})\backslash(\mathrm{Vars}(\bar{v})\backslash\mathrm{Vars}(\bar{z}))$. Then

$$\mathrm{Vars}(\bar{x})\backslash\mathrm{Vars}(\bar{y}) = \mathrm{Vars}(\bar{x}) \cap (\mathrm{Vars}(\bar{v})\backslash\mathrm{Vars}(\bar{z})) \subseteq (\mathrm{Vars}(\bar{v})\backslash\mathrm{Vars}(\bar{z})).$$

Therefore, every variable from $\mathrm{Vars}(\bar{x})\backslash\mathrm{Vars}(\bar{y})$ is by induction referential in p and thus in q. As a consequence:

$$\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); r. \tag{4.33}$$

Indeed:

$$\begin{aligned}
p &\doteq \mathrm{do}\ \bar{x} \leftarrow q; r \\
&= \mathrm{do}\ \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); \mathrm{ret}\,\bar{x}); r && \text{[by ind.]} \\
&= \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); r && \text{[by Lem. 4.1, } \textbf{(unit}_\mathbf{2}\textbf{)}]
\end{aligned}$$

Then the induction step is proven as follows:

$$\begin{aligned}
p &= \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); r && \text{[by 4.33]} \\
&= \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by ind.]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by Lem. 4.1]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q; \mathrm{ret}\,\bar{y}); r); \mathrm{ret}\,\bar{z}); \\
&\quad\ \mathrm{ret}\,\bar{v} && \text{[by Lem. 4.1]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v} && \text{[by 4.33]}
\end{aligned}$$

FL4. Let $p \doteq (\mathrm{init}\,\bar{v} \leftarrow q\,\mathrm{in}\,r^\star)$. By the assumption, the variables from $\mathrm{Vars}(\bar{v})\backslash\mathrm{Vars}(\bar{z})$ are referential in p and therefore also in q and r. Let us put by definition $u := (\mathrm{init}\,\bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z})\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z})^\star)$ and show that

$$\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{z} \leftarrow u; \mathrm{ret}\,\bar{v}. \tag{4.34}$$

To that end we shall make use of the identity

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,r^\star = \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z})^\star\ \mathrm{res}\,\bar{z} \to \mathrm{ret}\,\bar{v} \tag{4.35}$$

which follows from the evidently provable identity

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow \mathrm{ret}\,\bar{v}; r = \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v}$$

by Corollary 4.8. Then we have:

$$\begin{aligned}
p &\doteq \mathrm{init}\,\bar{v} \leftarrow q\,\mathrm{in}\,r^\star \\
&= \mathrm{init}\,\bar{v} \leftarrow (\mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z}); \mathrm{ret}\,\bar{v})\,\mathrm{in}\,r^\star && \text{[by ind.]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z}); \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,r^\star && \text{[by Lem. 4.1]} \\
&= \mathrm{do}\ \bar{z} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z}); \\
&\quad\ \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow r; \mathrm{ret}\,\bar{z})^\star\ \mathrm{res}\,\bar{z} \to \mathrm{ret}\,\bar{v} && \text{[by 4.35]}
\end{aligned}$$

$$\begin{aligned} &= \mathsf{init}\,\bar z \leftarrow (\mathsf{do}\ \bar v \leftarrow q; \mathsf{ret}\,\bar z)\,\mathsf{in}(\mathsf{do}\ \bar v \leftarrow r; \mathsf{ret}\,\bar z)^\star \\ &\quad \mathsf{res}\,\bar z \to \mathsf{ret}\,\bar v && \text{[by Lem. 4.1]} \\ &\doteq \mathsf{do}\ \bar z \leftarrow u; \mathsf{ret}\,\bar v \end{aligned}$$

Then the induction step can be proven as follows:

$$\begin{aligned} p &= \mathsf{do}\ \bar z \leftarrow u; \mathsf{ret}\,\bar v && \text{[by 4.34]} \\ &= \mathsf{do}\ \bar z \leftarrow (\mathsf{do}\ \bar v \leftarrow (\mathsf{do}\ \bar z \leftarrow q; \mathsf{ret}\,\bar v); \mathsf{ret}\,\bar z); \mathsf{ret}\,\bar v && \text{[by Lem. 4.1]} \\ &= \mathsf{do}\ \bar z \leftarrow (\mathsf{do}\ \bar v \leftarrow p; \mathsf{ret}\,\bar z); \mathsf{ret}\,\bar v && \text{[by 4.34]} \end{aligned}$$

The proof of the lemma is thus completed. □

Informally, Lemma 4.26 states that referential variables can be cut harmlessly.

Lemma 4.27. *Suppose we are given a $\bar z$-flat program p and a variable v not occurring in p. Let $z \in \mathsf{Vars}(p)$ and let $\bar v := \bar z[v/z]$. Then there exist effectively computable programs p_1 and p_2 such that*

1. $\vdash_{\mathsf{ME}^\star} p[v/z] = p_1 + p_2$,
2. $z \notin \mathsf{Vars}(p_1)$, $z \notin \mathsf{Vars}(p_2)$,
3. *p_1 is $\bar z$-flat, p_2 is $\bar v$-flat,*
4. *every variable referential in p is referential both in p_1 and in p_2,*
5. *if p is cut-free then both p_1 and p_2 are cut-free.*

Proof. By Remark 4.21 we ensure that Kleene star occurs in p only in the form $(\mathsf{init}\,\bar x \leftarrow \mathsf{ret}\,\bar x\,\mathsf{in}\,r^\star)$. Then we construct the programs p_1, p_2 in question by induction over term complexity of p with respect to the clauses of Definition 4.13.

FL1. We define p_1 and p_2 according to the specific form of p as follows.

If $p \doteq a_{i \triangleright \bar z}$ and $z = x_i$ then $p_1 := (a[v/z])_{i \triangleright \bar z}$ and $p_2 := \varnothing$.

If $p \doteq a_{i \triangleright \bar z}$ and either $z \neq x$ or $i = 0$ then $p_1 := \varnothing$ and $p_2 := (a[v/z])_{i \triangleright \bar v}$.

If $p \doteq \mathsf{ret}\,\bar z$ then $p_1 := \varnothing$ and $p_2 = \mathsf{ret}\,\bar v$.

If $p \doteq \varnothing$ then $p_1 := \varnothing$ and $p_2 := \varnothing$.

The induction invariant is easily verified.

FL2. Let $p \doteq (q + r)$. By induction hypothesis there exist q_1, q_2 and r_1, r_2 satisfying the induction invariant for q and r. We put $p_i := q_i + r_i$ for $i = 1, 2$. The remainder is then easily proven by the induction hypothesis.

FL3. Let $p \doteq (\text{do } \bar{x} \leftarrow q; r)$. Suppose, $z \notin \text{Vars}(\bar{x})$. By induction hypothesis there are q_1 and q_2 satisfying the induction invariant for q and there are r_1, r_2 satisfying the induction invariant for r. Note that since $\bar{x}[v/z] \doteq \bar{x}$, both q_1 and q_2 are $\bar{x}$-flat. Let us put by definition:

$$\begin{aligned} p_1 &:= \text{do } \bar{x} \leftarrow (q_1 + q_2); r_1, \\ p_2 &:= \text{do } \bar{x} \leftarrow (q_1 + q_2); r_2. \end{aligned}$$

Obviously, $p[v/z] \doteq (\text{do } \bar{z} \leftarrow q[v/z]; r[v/z])$ is provably equal to $p_1 + p_2$. By induction, every variable that is referential in p is referential in p_1, p_2 and $z \notin \text{Vars}(p_1), z \notin \text{Vars}(p_2)$.

Suppose p is cut-free, and prove that p_1, p_2 are cut-free. By the definition of cut-free, q and r are cut-free and every $x \notin \text{Vars}(\bar{z})$ is referential in q. By induction, q_1, q_2, r_1, r_2 are cut-free and every $x \notin \text{Vars}(\bar{z})$ is referential in q_1, q_2. This immediately implies that p_1 is cut-free. In order to prove that p_2 is also cut-free, we are left to show that every $x \notin \text{Vars}(\bar{v}) = \text{Vars}(\bar{z}[v/z])$ is referential in q_1, q_2. If $x \notin \text{Vars}(\bar{z})$ then we are done. Otherwise, $x = z$. By assumption of the lemma, x is referential in q. Therefore, by induction it is referential both in q_1 and in q_2, and we are done.

Suppose $z \in \text{Vars}(\bar{x})$. Let q_1, q_2 and r_1, r_2 be the programs satisfying by the induction invariant for q and r correspondingly. We define:

$$\begin{aligned} p_1 &:= \text{do } \bar{x} \leftarrow q_1; r + \text{do } \bar{y} \leftarrow q_2; r_1 \\ p_2 &:= \text{do } \bar{y} \leftarrow q_2; r_2 \end{aligned}$$

where $\bar{y} := \bar{x}[v/z]$. Let us prove that both p_1 and p_2 satisfy the induction invariant for p. Obviously p_1 is $\bar{z}$-flat, p_2 is $\bar{v}$-flat. It easily follows by induction that $z \notin \text{Vars}(p_1), z \notin \text{Vars}(p_2)$ (notice that the possible occurrences of z in the subterm r of p_1 are bound since $z \in \text{Vars}(\bar{x})$). Let some variable x be referential in p. Then it is referential both in q and r. By induction, x is referential in q_1, q_2, r_1, r_2 and thus it is referential in p_1, p_2. The following calculation proves $\vdash_{\mathsf{ME}^\star} p[v/z] = p_1 + p_2$:

$$\begin{aligned} p[v/z] &\equiv \text{do } \bar{x} \leftarrow q[v/z]; r \\ &= \text{do } \bar{x} \leftarrow (q_1 + q_2); r && \text{[by ind.]} \\ &= \text{do } \bar{x} \leftarrow q_1; r + \text{do } \bar{y} \leftarrow q_2; r[v/z] && \text{[by } \mathbf{(dist_1^+)}\text{]} \\ &= \text{do } \bar{x} \leftarrow q_1; r + \text{do } \bar{y} \leftarrow q_2; r_1 + \text{do } \bar{v} \leftarrow q_2; r_2 && \text{[by ind.]} \\ &= p_1 + p_2 && \text{[by def. of } p_i\text{]} \end{aligned}$$

Finally, suppose that p is cut-free, and prove that p_1, p_2 are too. By the definition of cut-free, q, r are cut-free and every $x \notin \mathrm{Vars}(\bar{z})$ is referential in q. By induction, q_1, q_2, r_1, r_2 are cut-free and x is referential in q_1, q_2. As an immediate consequence, p_1 is cut-free. In order to prove that p_2 is also cut-free, observe that since $z \notin \mathrm{Vars}(\bar{y})$ it is referential in q. Therefore, every $x \notin \mathrm{Vars}(\bar{v}) = \mathrm{Vars}(\bar{z}[v/z])$ is referential in q and thus, by induction, x is referential in q_2, i.e. p_2 is cut-free.

FL4. Let $p \doteq (\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}\,r^\star)$. First assume that $z \notin \mathrm{Vars}(\bar{z})$. This assumption immediately implies identity of $\bar{z}$ and $\bar{v}$. Let r_1 and r_2 be the $\bar{z}$-flat programs corresponding to r by induction hypothesis. We put $p_1 := (\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(r_1 + r_2)^\star)$, $p_2 := \varnothing$. It is easily verified that p_1, p_2 satisfy the induction invariant.

Suppose $z \in \mathrm{Vars}(\bar{z})$. Let r_1, r_2 be the programs provided by the induction invariant for r. Recall that r_1 is $\bar{z}$-flat and r_2 is $\bar{v}$-flat. Then we define

$$\begin{aligned} p_1 &:= \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,r_2^\star\,\mathrm{res}\,\bar{v} \to (\mathrm{init}\,\bar{z} \leftarrow r_1\,\mathrm{in}\,r^\star), \\ p_2 &:= \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,r_2^\star. \end{aligned}$$

By induction: $z \notin \mathrm{Vars}(p_1)$ and $z \notin \mathrm{Vars}(p_2)$. Let us show that $p[v/z]$ is provably equal to $p_1 + p_2$. Observe that:

$$\begin{aligned} &\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{z} \leftarrow \mathrm{ret}\,\bar{z}; r = \mathrm{do}\ \bar{z} \leftarrow r; \mathrm{ret}\,\bar{z}, \\ &\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{z} \leftarrow \mathrm{ret}\,\bar{v}; r = \mathrm{do}\ \bar{z} \leftarrow r_1; \mathrm{ret}\,\bar{z} + \mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{v}. \end{aligned}$$

We would like to match these equalities with the family of equalities in the premise of Lemma 4.6. To that end, we first need to prove that

$$(2, \{1 \mapsto \bar{z}, 2 \mapsto \bar{v}\}, \{(1,1) \mapsto r, (1,2) \mapsto \varnothing, (2,1) \mapsto r_1, (2,2) \mapsto r_2\})$$

is a computational net. Indeed, we have:

$$\begin{aligned} \mathrm{Vars}(\bar{v}) \cap \mathrm{Vars}(r) &\subseteq (\mathrm{Vars}(\bar{z}) \cap \mathrm{Vars}(r)) \cup (\{v\} \cap \mathrm{Vars}(r)) \\ &\subseteq \mathrm{Vars}(\bar{z}) \cup \emptyset = \mathrm{Vars}(\bar{z}), \\ \mathrm{Vars}(\bar{z}) \cap \mathrm{Vars}(r_i) &\subseteq (\mathrm{Vars}(\bar{v}) \cap \mathrm{Vars}(r_i)) \cup (\{z\} \cap \mathrm{Vars}(r_i)) \\ &\subseteq \mathrm{Vars}(\bar{v}) \cup \emptyset = \mathrm{Vars}(\bar{v}), \\ \mathrm{Vars}(\bar{v}) \cap \mathrm{Vars}(\varnothing) &= \emptyset \subseteq \mathrm{Vars}(\bar{z}) \end{aligned}$$

where $i = 1, 2$. The remaining cases are trivial. In a similar fashion, one can make sure that the further conditions of Lemma 4.6 are also met. We can thus conclude that the following equation is provable in $\mathsf{ME}^\star$:

$$\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,r^\star = \mathrm{do}\ \bar{v} \leftarrow w^2_{1,1}; \mathrm{ret}\,\bar{v} + \mathrm{do}\ \bar{z} \leftarrow w^2_{1,2}; \mathrm{ret}\,\bar{z} \tag{4.36}$$

where

$$\begin{aligned}
w^2_{1,1} &\doteq w^1_{1,1} + \mathsf{init}\,\bar{z} \leftarrow w^1_{1,2}\,\mathsf{in}(w^1_{2,2})^\star\,\mathsf{res}\,\bar{z} \to w^1_{2,1},\\
w^2_{1,2} &\doteq w^1_{1,2} + \mathsf{init}\,\bar{z} \leftarrow w^1_{1,2}\,\mathsf{in}(w^1_{2,2})^\star\,\mathsf{res}\,\bar{z} \to w^1_{2,2},\\
w^1_{1,1} &\doteq w^0_{1,1} + \mathsf{init}\,\bar{v} \leftarrow w^0_{1,1}\,\mathsf{in}(w^0_{1,1})^\star\,\mathsf{res}\,\bar{v} \to w^0_{1,1},\\
w^1_{2,2} &\doteq w^0_{2,2} + \mathsf{init}\,\bar{v} \leftarrow w^0_{1,1}\,\mathsf{in}(w^0_{1,1})^\star\,\mathsf{res}\,\bar{v} \to w^0_{1,2},\\
w^1_{1,2} &\doteq w^0_{1,2} + \mathsf{init}\,\bar{v} \leftarrow w^0_{1,1}\,\mathsf{in}(w^0_{1,1})^\star\,\mathsf{res}\,\bar{v} \to w^0_{1,2},\\
w^1_{2,1} &\doteq w^0_{2,1} + \mathsf{init}\,\bar{v} \leftarrow w^0_{2,1}\,\mathsf{in}(w^0_{1,1})^\star\,\mathsf{res}\,\bar{v} \to w^0_{1,1},
\end{aligned}$$

$$\begin{aligned}
w^0_{1,1} &\doteq \mathsf{ret}\,\bar{v} + r_2, & w^0_{1,2} &\doteq r_1,\\
w^0_{2,2} &\doteq \mathsf{ret}\,\bar{v} + r, & w^0_{2,1} &\doteq \varnothing.
\end{aligned}$$

After elementary simplifications in $\mathsf{ME}^\star$, we obtain:

$$\begin{aligned}
w^1_{2,1} &= \varnothing,\\
w^1_{1,2} &= \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,v\,\mathsf{in}\,r_2^\star\,\mathsf{res}\,\bar{v} \to r_1,\\
w^1_{2,2} &= \mathsf{ret}\,\bar{z} + r,\\
w^1_{1,1} &= \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r_2^\star,\\
w^2_{1,2} &= \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r_2^\star\,\mathsf{res}\,\bar{v} \to (\mathsf{init}\,\bar{z} \leftarrow r_1\,\mathsf{in}\,r^\star),\\
w^2_{1,1} &= \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r_2^\star.
\end{aligned}$$

The equality $p[v/z] = p_1 + p_2$ can now be established as follows:

$$\begin{aligned}
p[v/z] &\doteq \mathsf{init}\,\bar{z} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r^\star\\
&= \mathsf{do}\ \bar{v} \leftarrow w^2_{1,1}; \mathsf{ret}\,\bar{v} + \mathsf{do}\ \bar{z} \leftarrow w^2_{1,2}; \mathsf{ret}\,\bar{z} && \text{[by 4.36]}\\
&= w^2_{1,1} + w^2_{1,2} && \text{[by } \textbf{(unit}_1\textbf{)}]\\
&= \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r_2^\star +\\
&\quad\ \mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}\,r_2^\star\,\mathsf{res}\,\bar{v} \to (\mathsf{init}\,\bar{z} \leftarrow r_1\,\mathsf{in}\,r^\star)\\
&\equiv p_1 + p_2
\end{aligned}$$

Every variable that is referential in p is referential in r. By induction, it is referential in r_1, r_2 and thus by definition, it is referential in p_1, p_2. In the same manner, p being cut-free implies that p_1, p_2 are also cut-free. □

Lemma 4.28. *Let p be a $\bar{z}$-flat program. Then there exists an effectively computable cut-free $\bar{v}$-flat program q such that* $\mathrm{Vars}(\bar{z}) \subseteq \mathrm{Vars}(\bar{v})$ *and*

$$\vdash_{\mathsf{ME}^\star} p = \mathsf{do}\ \bar{v} \leftarrow q; \mathsf{ret}\,\bar{z}. \tag{4.37}$$

Proof. By Remark 4.21 we ensure that Kleene star occurs in p only as $(\mathsf{init}\,\bar{x} \leftarrow \mathsf{ret}\,\bar{x}\,\mathsf{in}\,r^\star)$. Then we proceed by induction over term complexity of p with respect to the clauses of Definition 4.13.

FL1. For the programs matching this clause, we put $q := p$. The induction invariant is trivially satisfied.

FL2. Let $p \doteq u + r$. By induction hypothesis there exist cut-free $\bar{v}$-flat programs u' and r' such that $\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{v} \leftarrow u'; \mathsf{ret}\,\bar{z} = u$ and $\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{v} \leftarrow r'; \mathsf{ret}\,\bar{z} = r$. Evidently $q := u' + r'$ proves the induction step.

FL3. Let $p \doteq (\mathsf{do}\ \bar{x} \leftarrow u; r)$. Let r' and u' be the cut-free programs ensured by the induction hypothesis such that r' is $\bar{z}'$-flat, u' is $\bar{x}'$-flat and

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{x}' \leftarrow u'; \mathsf{ret}\,\bar{x} = u,$$
$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{z} = r.$$

Let $\bar{x}^\blacktriangledown$ be a vector of distinct variables such that $\mathrm{Vars}(\bar{x}^\blacktriangledown) = \mathrm{Vars}(\bar{x}')\backslash\mathrm{Vars}(\bar{x})$ and let $\bar{x}^\vartriangle$ be the vector of fresh distinct variables of the same length. Let w be the program obtained from u' by replacing every $x_i^\blacktriangledown$ by $x_i^\vartriangle$ throughout. Then $w[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle] \equiv u'$ and

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{v} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow w[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v}); \mathsf{ret}\,\bar{z} = p \tag{4.38}$$

where $\bar{y} := \bar{x}'[\bar{x}^\vartriangle/\bar{x}^\blacktriangledown]$ and $\bar{v}$ is such that $\mathrm{Vars}(\bar{v}) = \mathrm{Vars}(\bar{z}') \cup \mathrm{Vars}(\bar{x}^\vartriangle) \cup \mathrm{Vars}(\bar{x}^\blacktriangledown)$. Indeed:

$$\begin{aligned}
&\mathsf{do}\ \bar{v} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow w[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v}); \mathsf{ret}\,\bar{z} \\
&\quad = \mathsf{do}\ \bar{v} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow u'; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v}); \mathsf{ret}\,\bar{z} \\
&\quad = \mathsf{do}\ \bar{y} \leftarrow u'; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{z} && \text{[by Lemma 4.1]} \\
&\quad = \mathsf{do}\ \bar{y} \leftarrow u'; r && \text{[by ind.]} \\
&\quad = \mathsf{do}\ \bar{x} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow u'; \mathsf{ret}\,\bar{x}); r && \text{[by Lemma 4.1]} \\
&\quad = \mathsf{do}\ \bar{x} \leftarrow (\mathsf{do}\ \bar{x}' \leftarrow u'; \mathsf{ret}\,\bar{x}); r \\
&\quad = \mathsf{do}\ \bar{x} \leftarrow u; r. && \text{[by ind.]}
\end{aligned}$$

We will be done once we manage to reduce $(\mathsf{do}\ \bar{y} \leftarrow w[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v})$ to a cut-free $\bar{v}$-flat program. We proceed by further induction over $n := |\bar{x}^\blacktriangledown|$. If $n = 0$ then we are done by outer induction. Suppose $n > 0$. By Lemma 4.27, $w[\bar{x}_n^\blacktriangledown/\bar{x}_n^\vartriangle]$ is provably equal to a sum $w_1 + w_2$ where w_1 is $\bar{y}$-flat and w_2 is $\bar{y}[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]$-flat. Then

$$\begin{aligned}
&\mathsf{do}\ \bar{y} \leftarrow w[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v} \\
&\quad = \mathsf{do}\ \bar{y} \leftarrow w_1[\bar{x}_{\hat{n}}^\blacktriangledown/\bar{x}_{\hat{n}}^\vartriangle]; \bar{z}' \leftarrow r'; \mathsf{ret}\,\bar{v} +
\end{aligned}$$

$$\mathsf{do}\ \bar y \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \bar z' \leftarrow r'; \mathsf{ret}\,\bar v.$$

By induction, first summand of the latter sum is provably equal to a $\bar v$-flat cut-free program. We will be done as soon as we prove that the same is true for the latter summand. Let $\bar y' := \bar y[\bar x^{\blacktriangledown}_{n}/\bar x^{\vartriangle}_{n}]$, or equivalently, $\bar y' := \bar x'[\bar x^{\vartriangle}_{\hat n}/\bar x^{\blacktriangledown}_{\hat n}]$. Let y'' be a vector of distinct variables such that $\mathrm{Vars}(\bar y'') = \mathrm{Vars}(\bar y)\backslash\{x^{\vartriangle}_{n}\}$. Then $\mathrm{Vars}(\bar y'') = \mathrm{Vars}(\bar y')\backslash\{x^{\blacktriangledown}_{n}\}$. Note that $x^{\vartriangle}_{n}$ is referential in w_2 and thus also in $w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]$. Therefore we have:

$$\begin{aligned}
&\mathsf{do}\ \bar y \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \bar z' \leftarrow r'; \mathsf{ret}\,\bar v\\
&\quad = \mathsf{do}\ \bar y'' \leftarrow (\mathsf{do}\ \bar y \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \mathsf{ret}\,\bar y''); \bar z' \leftarrow r'; \mathsf{ret}\,\bar v && \text{[by Lem. 4.1]}\\
&\quad = \mathsf{do}\ \bar y'' \leftarrow (\mathsf{do}\ \bar y' \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \mathsf{ret}\,\bar y''); \bar z' \leftarrow r'; \mathsf{ret}\,\bar v\\
&\quad = \mathsf{do}\ \bar y' \leftarrow (\mathsf{do}\ \bar y'' \leftarrow (\mathsf{do}\ \bar y' \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \mathsf{ret}\,\bar y''); \mathsf{ret}\,\bar y');\\
&\qquad \bar z' \leftarrow r'; \mathsf{ret}\,\bar v && \text{[by Lem. 4.1]}\\
&\quad = \mathsf{do}\ \bar y' \leftarrow w_2[\bar x^{\blacktriangledown}_{\hat n}/\bar x^{\vartriangle}_{\hat n}]; \bar z' \leftarrow r'; \mathsf{ret}\,\bar v && \text{[by Lem. 4.26]}
\end{aligned}$$

and we are done by inner induction hypothesis.

FL4. Let $p \doteq (\mathsf{init}\,\bar z \leftarrow \mathsf{ret}\,z\,\mathsf{in}\,u^\star)$. According to the induction hypothesis we assume that there is some $\bar z'$-flat cut-free program u' such that

$$\vdash_{\mathsf{ME}^\star} u = \mathsf{do}\ \bar z' \leftarrow u'; \mathsf{ret}\,\bar z \tag{4.39}$$

and $\mathrm{Vars}(\bar z) \subseteq \mathrm{Vars}(\bar z')$. Let $\bar z^{\blacktriangledown}$ be a vector of distinct variables such that $\mathrm{Vars}(\bar z^{\blacktriangledown}) = \mathrm{Vars}(\bar z')\backslash\mathrm{Vars}(\bar z)$ and let $\bar z^{\vartriangle}$ be the vector of fresh distinct variables of the same length. Let w be the program obtained from u' by replacing every $z^{\blacktriangledown}_i$ by $z^{\vartriangle}_i$ throughout. Then $w[\bar z^{\blacktriangledown}/\bar z^{\vartriangle}] \equiv u'$,

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar v \leftarrow (\mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}(\mathsf{do}\ \bar y \leftarrow w[\bar z^{\blacktriangledown}/\bar z^{\vartriangle}]; \mathsf{ret}\,\bar v)^\star); \mathsf{ret}\,\bar z = p \tag{4.40}$$

where $\bar y := \bar z'[\bar z^{\vartriangle}/\bar z^{\blacktriangledown}]$ and $\bar v$ is such that $\mathrm{Vars}(\bar v) = \mathrm{Vars}(\bar z') \cup \mathrm{Vars}(\bar z^{\vartriangle})$. We have:

$$\begin{aligned}
&\mathsf{do}\ \bar v \leftarrow (\mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}(\mathsf{do}\ \bar y \leftarrow w[\bar z^{\blacktriangledown}/\bar z^{\vartriangle}]; \mathsf{ret}\,\bar v)^\star); \mathsf{ret}\,\bar z\\
&\qquad = \mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}(\mathsf{do}\ \bar y \leftarrow u'; \mathsf{ret}\,\bar v)^\star\ \mathsf{res}\,\bar v \to \mathsf{ret}\,\bar z\\
&\qquad = \mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}(\mathsf{do}\ \bar z \leftarrow (\mathsf{do}\ \bar y \leftarrow u'; \mathsf{ret}\,\bar z); \mathsf{ret}\,\bar v)^\star\\
&\qquad\quad \mathsf{res}\,\bar v \to \mathsf{ret}\,\bar z && \text{[by Lem. 4.1]}\\
&\qquad = \mathsf{init}\,\bar v \leftarrow \mathsf{ret}\,\bar v\,\mathsf{in}(\mathsf{do}\ \bar z \leftarrow (\mathsf{do}\ \bar z' \leftarrow u'; \mathsf{ret}\,\bar z); \mathsf{ret}\,\bar v)^\star\\
&\qquad\quad \mathsf{res}\,\bar v \to \mathsf{ret}\,\bar z.
\end{aligned}$$

Observe that by Lemma 4.1 and **(unit$_2$)**:

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar z \leftarrow \mathsf{ret}\,\bar z; (\mathsf{do}\ \bar z' \leftarrow u'; \mathsf{ret}\,\bar z)$$

$$= \mathsf{do}\ \bar{v} \leftarrow (\mathsf{do}\ \bar{z} \leftarrow (\mathsf{do}\ \bar{z}' \leftarrow u'; \mathsf{ret}\,\bar{z}); \mathsf{ret}\,\bar{v}); \mathsf{ret}\,\bar{z}$$

and thus we can complete the calculation by Corollary 4.8:

$$\begin{aligned} &\mathsf{init}\,\bar{v} \leftarrow \mathsf{ret}\,\bar{v}\,\mathsf{in}(\mathsf{do}\ \bar{z} \leftarrow (\mathsf{do}\ \bar{z}' \leftarrow u'; \mathsf{ret}\,\bar{z}); \mathsf{ret}\,\bar{v})^\star\ \mathsf{res}\,\bar{v} \to \mathsf{ret}\,\bar{z} \\ &\quad = \mathsf{init}\,\bar{z} \leftarrow \mathsf{ret}\,\bar{z}\,\mathsf{in}(\mathsf{do}\ \bar{z}' \leftarrow u'; \mathsf{ret}\,\bar{z})^\star && [\text{Cor. } 4.8] \\ &\quad = \mathsf{init}\,\bar{z} \leftarrow \mathsf{ret}\,\bar{z}\,\mathsf{in}\,u^\star. && [\text{by } 4.39] \end{aligned}$$

We show that $(\mathsf{do}\ \bar{y} \leftarrow w[\bar{z}^\blacktriangledown/\bar{z}^\vartriangle]; \mathsf{ret}\,\bar{v})$ is provably equal to an effectively computable $\bar{v}$-flat cut-free program. Since $\mathrm{Vars}(\bar{z}) \subseteq \mathrm{Vars}(\bar{z}') \subseteq \mathrm{Vars}(\bar{v})$, by (4.40) this would imply the claim. We proceed by induction over $n := |\bar{z}^\blacktriangledown|$. If $n = 0$, i.e. $\mathrm{Vars}(\bar{z}) = \mathrm{Vars}(\bar{z}') = \mathrm{Vars}(\bar{v})$ then we are trivially done. Suppose $n > 0$. By Lemma 4.27, $w[z_n^\blacktriangledown/z_n^\vartriangle]$ is provably equal to a sum $w_1 + w_2$ where w_1 is $\bar{y}$-flat and w_2 is $\bar{y}[z_n^\blacktriangledown/z_n^\vartriangle]$-flat. Then

$$\begin{aligned} &\mathsf{do}\ \bar{y} \leftarrow w[\bar{z}^\blacktriangledown/\bar{z}^\vartriangle]; \mathsf{ret}\,\bar{v} \\ &\quad = \mathsf{do}\ \bar{y} \leftarrow w_1[\bar{z}_{\hat{n}}^\blacktriangledown/\bar{z}_{\hat{n}}^\vartriangle]; \mathsf{ret}\,\bar{v} + \mathsf{do}\ \bar{y} \leftarrow w_2[\bar{z}_{\hat{n}}^\blacktriangledown/\bar{z}_{\hat{n}}^\vartriangle]; \mathsf{ret}\,\bar{v}. \end{aligned}$$

Like in the previous clause, first summand of the latter sum is provably equal to a $\bar{v}$-flat cut-free program. We will be done as soon as we show the same for the second summand. Let $\bar{y}' := \bar{y}[z_n^\blacktriangledown/z_n^\vartriangle]$, or equivalently, $\bar{y}' := \bar{z}'[\bar{z}_{\hat{n}}^\vartriangle/\bar{z}_{\hat{n}}^\blacktriangledown]$. Let y'' be a vector of distinct variables such that $\mathrm{Vars}(\bar{y}'') = \mathrm{Vars}(\bar{y})\backslash\{z_n^\vartriangle\}$. Then $\mathrm{Vars}(\bar{y}'') = \mathrm{Vars}(\bar{y}')\backslash\{z_n^\blacktriangledown\}$. Note that $z_n^\vartriangle$ is referential in w_2, and thus it is also referential in $w_2[\bar{z}_{\hat{n}}^\blacktriangledown/\bar{z}_{\hat{n}}^\vartriangle]$. Then

$$\begin{aligned} &\mathsf{do}\ \bar{y} \leftarrow w_2[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \mathsf{ret}\,\bar{v} \\ &\quad = \mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{y}'' \leftarrow (\mathsf{do}\ \bar{y} \leftarrow w_2[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \\ &\qquad \mathsf{ret}\,\bar{y}''); \mathsf{ret}\,\bar{y}); \mathsf{ret}\,\bar{v} && [\text{by Lem. } 4.26] \\ &\quad = \mathsf{do}\ \bar{y}'' \leftarrow (\mathsf{do}\ \bar{y} \leftarrow w_2[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \mathsf{ret}\,\bar{y}''); \mathsf{ret}\,\bar{v} && [\text{by Lem. } 4.1] \\ &\quad = \mathsf{do}\ \bar{y}'' \leftarrow (\mathsf{do}\ \bar{y}' \leftarrow w_2[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \mathsf{ret}\,\bar{y}''); \mathsf{ret}\,\bar{v} \\ &\quad = \mathsf{do}\ \bar{y}' \leftarrow w_2[\bar{x}^\blacktriangledown/\bar{x}^\vartriangle]; \mathsf{ret}\,\bar{v} && [\text{by Lem. } 4.1] \end{aligned}$$

and we are done by inner induction hypothesis. □

Lemma 4.29. *Let p be a $\bar{z}$-flat cut-free program. For every vector of distinct variables $\bar{v}$ there is an effectively found $\bar{v}$-flat tight program q such that $\vdash_{\mathsf{ME}^\star}\ q = \mathsf{do}\ \bar{z} \leftarrow p; \mathsf{ret}\,\bar{v}$.*

Proof. First, we ensure by Remark 4.21 that Kleene star occurs in p only in the form $(\mathsf{init}\,\bar{x} \leftarrow \mathsf{ret}\,\bar{x}\,\mathsf{in}\,r^\star)$. Then we construct by induction over term complexity of p a vector of tight $\bar{v}$-flat programs $\bar{p}$ such that

$$\vdash_{\mathsf{ME}^\star}\ \mathsf{do}\ \bar{z} \leftarrow p; \mathsf{ret}\,\bar{v} = \textstyle\sum_i p_i \tag{4.41}$$

and for every p_i and every $t \in \mathsf{tr}(p_i)$, $\mathrm{Vars}(t) = \mathrm{Vars}(p_i) \subseteq \mathrm{Vars}(p) \cup (\mathrm{Vars}(\bar{v}) \backslash \mathrm{Vars}(\bar{z}))$. Let us proceed by case distinction over the clauses of Definition 4.13.

FL1. If $p \doteq \varnothing$, then $\bar{p} := \star$. If $p \doteq \mathsf{ret}\,\bar{z}$, then $\bar{p} := \mathsf{ret}\,\bar{v}$. If $p \doteq a_{i \triangleright \bar{z}}$, then $\bar{p} := a_{i \triangleright \bar{v}}$. The induction invariant is trivially verified.

FL2. Let $p \doteq (u + r)$. Then, by induction, there exist vectors of $\bar{z}$-flat programs $\bar{u}$ and $\bar{r}$ such that $\vdash_{\mathsf{ME}^\star}$ $\mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v}$, $\vdash_{\mathsf{ME}^\star}$ $\mathsf{do}\ \bar{z} \leftarrow u; \mathsf{ret}\,\bar{v}$. We define $\bar{p} := \langle \bar{u}, \bar{r} \rangle$. The induction invariant is clearly satisfied.

FL3. Let $p \doteq (\mathsf{do}\ \bar{x} \leftarrow u; r)$. Let $\bar{r}$ be the sequence of $\bar{v}$-flat tight program such that $\vdash_{\mathsf{ME}^\star}$ $\mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v} = \sum_j r_j$ whose existence is guaranteed by the induction hypothesis. For every j, let $\bar{y}^j$ be a vector of variables such that $\mathrm{Vars}(\bar{y}^j) = \mathrm{Vars}(r_j)$, and let $\bar{u}^j$ be the sequence of $\bar{y}^j$-flat tight programs provided by induction hypothesis, i.e. such that for every j, $\vdash_{\mathsf{ME}^\star}$ $\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y}^j = \sum_k u^j_k$.

By definition, every $w_{j,k} := (\mathsf{do}\ \bar{y}^j \leftarrow u^j_k; r_j)$ is $\bar{v}$-flat. Let $\bar{p}$ be a vector consisting of all the $w_{j,k}$ arranged arbitrarily. Let us verify the induction invariant. By induction, both u^j_k and r_j are tight and for every $t \in \mathsf{tr}(r_j)$, $\mathrm{Vars}(t) = \mathrm{Vars}(r_j) = \mathrm{Vars}(\bar{y}^j)$. Therefore, $w_{j,k}$ is also tight. For every $t \in \mathsf{tr}(w_{j,k})$ there is $s \in \mathsf{tr}(u^j_k)$ such that $\mathrm{Vars}(t) = \mathrm{Vars}(s)$. By induction, $\mathrm{Vars}(s) = \mathrm{Vars}(u^j_k)$ and by tightness of $w_{j,k}$, $\mathrm{Vars}(u^j_k) = \mathrm{Vars}(w_{j,k})$. We have thus proved that for every $t \in \mathsf{tr}(w_{j,k})$, $\mathrm{Vars}(t) = \mathrm{Vars}(w_{j,k})$. Moreover, by induction:

$$\begin{aligned}
\mathrm{Vars}(w_{j,k}) &\subseteq \mathrm{Vars}(u^j_k) \\
&\subseteq \mathrm{Vars}(u) \cup (\mathrm{Vars}(\bar{y}^j) \backslash \mathrm{Vars}(\bar{x})) = \mathrm{Vars}(u) \cup (\mathrm{Vars}(r_j) \backslash \mathrm{Vars}(\bar{x})) \\
&\subseteq \mathrm{Vars}(u) \cup (\mathrm{Vars}(r) \backslash \mathrm{Vars}(\bar{x})) \cup ((\mathrm{Vars}(\bar{v}) \backslash \mathrm{Vars}(\bar{z})) \backslash \mathrm{Vars}(\bar{x})) \\
&\subseteq \mathrm{Vars}(p) \cup (\mathrm{Vars}(\bar{v}) \backslash \mathrm{Vars}(\bar{z})).
\end{aligned}$$

Finally, let us verify (4.41). Let $\bar{z}'$ be a vector of distinct variables satisfying the equation $\mathrm{Vars}(\bar{y}) = \mathrm{Vars}(\bar{z}) \cap \mathrm{Vars}(\bar{x})$. Then we have:

$$\begin{aligned}
\textstyle\sum_{j,k} w_{j,k} &= \textstyle\sum_j \sum_k \mathsf{do}\ \bar{y}^j \leftarrow u^j_k; r_j \\
&= \textstyle\sum_j \mathsf{do}\ \bar{y}^j \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y}^j); r_j && \text{[by ind.]} \\
&= \textstyle\sum_j \mathsf{do}\ \bar{x} \leftarrow u; r_j && \text{[by Lem. 4.1]} \\
&= \mathsf{do}\ \bar{x} \leftarrow u; \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v} && \text{[by ind.]} \\
&= \mathsf{do}\ \bar{x} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y}); \mathsf{ret}\,\bar{x}); \\
&\qquad \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v} && \text{[by Lem. 4.26]} \\
&= \mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y}); \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v} && \text{[by Lem. 4.1]} \\
&= \mathsf{do}\ \bar{z} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y}); r); \mathsf{ret}\,\bar{v} && \text{[by Lem. 4.1]} \\
&= \mathsf{do}\ \bar{z} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow (\mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; \mathsf{ret}\,\bar{y});
\end{aligned}$$

$$
\begin{aligned}
& \mathsf{ret}\,\bar{x}); r); \mathsf{ret}\,\bar{v} && \text{[by Lem. 4.1]}\\
&= \mathsf{do}\ \bar{z} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow u; r); \mathsf{ret}\,\bar{v}. && \text{[by Lem. 4.26]}
\end{aligned}
$$

which completes the proof of the clause.

FL4. Let $p \doteq (\mathsf{init}\,\bar{z} \leftarrow \mathsf{ret}\,\bar{z}\,\mathsf{in}\,r^\star)$. Let $\{\bar{v}^1, \ldots, \bar{v}^n\}$ be a finite set of vectors of distinct variables such that $\bar{v}^1 \doteq \bar{v}$ and $\{\mathsf{Vars}(\bar{v}^i)\}_i = \mathcal{P}(\mathsf{Vars}(p) \cup \mathsf{Vars}(\bar{v}))$. By induction hypothesis, for every i there exists a vector of tight $\bar{v}^i$-flat programs $\bar{p}^i$ such that

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v}^i = \textstyle\sum_j p^i_j. \tag{4.42}$$

Since for every i, j, $\mathsf{Vars}(p^i_j) \subseteq \mathsf{Vars}(p) \cup (\mathsf{Vars}(\bar{v}^i) \backslash \mathsf{Vars}(\bar{z})) \subseteq \mathsf{Vars}(p) \cup \mathsf{Vars}(\bar{v})$, we can assume that for every i, $|\bar{p}^i| = n$ and for every j, $\mathsf{Vars}(p^i_j) = \mathsf{Vars}(\bar{v}^j)$. Otherwise we ensure this property by replacing every two distinct programs from $\bar{p}^i$ with the same set of free variables by their sum, rearranging the elements of $\bar{p}^i$ and possibly inserting some deadlocks.

For all appropriate i, j, k, $\mathsf{Vars}(\bar{v}^k) \cap \mathsf{Vars}(p^i_j) = \mathsf{Vars}(\bar{v}^k) \cap \mathsf{Vars}(\bar{v}^j) \subseteq \mathsf{Vars}(\bar{v}^j)$ and hence $(n, \lambda i.\, \bar{v}^i, \lambda j.\, \lambda i.\, p^i_j)$ makes a computational net. Let $(n, \lambda i.\, \bar{v}^i, \lambda j.\, \lambda i.\, \hat{p}^i_j)$ be its transitive closure. Similarly, for all i, j, $\mathsf{Vars}(\bar{z}) \cap \mathsf{Vars}(p^i_j) \subseteq \mathsf{Vars}(\bar{v}^j)$. Finally, by **(unit₂)**, (4.42) can be rewritten as:

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v}^i = \textstyle\sum_j \mathsf{do}\ \bar{v}^j \leftarrow \mathsf{ret}\,\bar{v}^j; p^i_j.$$

Therefore, by Lemma (4.7), for every i:

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v}^i = \textstyle\sum_j \mathsf{do}\ \bar{v}^j \leftarrow \mathsf{ret}\,\bar{v}^j; \hat{p}^i_j.$$

In particular, if we instantiate i by 1 and perform simplification under **(unit₂)**, we obtain: $\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{z} \leftarrow r; \mathsf{ret}\,\bar{v} = \sum_j \hat{p}^1_j$. It can be easily shown by further induction that for every j, k, $\hat{p}^k_j$ is tight $\bar{v}$-flat and $\mathsf{Vars}(p^k_j) = \mathsf{Vars}(\bar{v}^j)$.

Let us return to the proof of the induction invariant. We define $\bar{p} := \langle \hat{p}^1_1, \ldots, \hat{p}^1_n \rangle$. Under this definition we obtain (4.41) straight away. We are left to show that for every j and every $t \in \mathsf{tr}(\hat{p}^1_j)$, $\mathsf{Vars}(t) = \mathsf{Vars}(\hat{p}^1_j) \subseteq \mathsf{Vars}(p) \cup (\mathsf{Vars}(\bar{v}) \backslash \mathsf{Vars}(\bar{z}))$. Let $t \in \mathsf{tr}(\hat{p}^1_j)$. Then by tightness, $\mathsf{Vars}(t) = \mathsf{Vars}(s)$ where $s \in \{\mathsf{ret}\,\bar{v}^j, p^1_j, \ldots, p^n_j\}$, i.e. $\mathsf{Vars}(t) = \mathsf{Vars}(\bar{v}^j) = \mathsf{Vars}(\hat{p}^1_j)$. Finally,

$$
\begin{aligned}
\mathsf{Vars}(\hat{p}^1_j) &= \mathsf{Vars}(\bar{v}^j) = \mathsf{Vars}(p^1_j)\\
&\subseteq \mathsf{Vars}(r) \cup (\mathsf{Vars}(\bar{v}) \backslash \mathsf{Vars}(\bar{z})) \subseteq \mathsf{Vars}(p) \cup (\mathsf{Vars}(\bar{v}) \backslash \mathsf{Vars}(\bar{z}))
\end{aligned}
$$

and thus we are done. □

In conjunction with Lemma 4.28 the latter result gives rise to

Corollary 4.30. *For every flat program p, there exists an effectively computable tight program q with the same footprint such that $\vdash_{\mathsf{ME}^\star} p = q$.*

Now we can prove the main result of this section.

Lemma 4.31. *Let p and q be two flat programs with the same footprint. Then the problems $\vdash_{\mathsf{ME}^\star} p = q$ and $\vdash_{\mathsf{ME}^\omega} p = q$ are equivalent and decidable.*

Proof. Let Λ be a name pool and let $\sigma : \mathrm{Vars}(p) \cup \mathrm{Vars}(q) \to \Lambda$ be an arbitrary injective substitution. We prove the statement of the lemma with p replaced by $p\sigma$ and q replaced by $q\sigma$. This would obviously imply the original formulation. Observe that by definition, $\vdash_{\mathsf{ME}^\star} p\sigma = q\sigma$ implies $\vdash_{\mathsf{ME}^\omega} p\sigma = q\sigma$. Let us assume that $\vdash_{\mathsf{ME}^\omega} p\sigma = q\sigma$ and prove $\vdash_{\mathsf{ME}^\star} p\sigma = q\sigma$. By Corollary 4.30 and Lemma 4.22, $\vdash_{\mathsf{ME}^\star} p\sigma = \sum_\varsigma p_{\sigma_p,\varsigma}$ and $\vdash_{\mathsf{ME}^\star} q\sigma = \sum_\varsigma q_{\sigma_q,\varsigma}$ where σ_p is the restriction of σ to $\mathrm{Vars}(p)$, σ_q is the restriction of σ to $\mathrm{Vars}(q)$ and ς ranges over injective substitutions from $\mathrm{Vars}(\bar{z})$ to some finite subset of Λ. We refer to the statement of Lemma 4.22 for the full list of properties satisfied by the programs $p_{\sigma_p,\varsigma}$ and $q_{\sigma_q,\varsigma}$.

At the moment we have established: $\vdash_{\mathsf{ME}^\omega} \sum_\varsigma p_{\sigma_p,\varsigma} = \sum_\varsigma q_{\sigma_q,\varsigma}$. Let $\bar{z}$ be the common footprint of p and q. We denote by $\Xi_{\bar{v}}$ the set of those ς for which $\bar{z}\varsigma \doteq \bar{v}$. Then we obviously have

$$\vdash_{\mathsf{ME}^\omega} \sum_{\bar{v}} \sum_{\varsigma\in\Xi_{\bar{v}}} p_{\sigma_p,\varsigma} = \sum_{\bar{v}} \sum_{\varsigma\in\Xi_{\bar{v}}} q_{\sigma_q,\varsigma}. \tag{4.43}$$

Let us prove that for all $\bar{v}$

$$\vdash_{\mathsf{ME}^\omega} \sum_{\varsigma\in\Xi_{\bar{v}}} p_{\sigma_p,\varsigma} = \sum_{\varsigma\in\Xi_{\bar{v}}} q_{\sigma_q,\varsigma}. \tag{4.44}$$

By Corollary 3.39 it suffices to show that $\bigcup_{\varsigma\in\Xi_{\bar{v}}} \mathsf{NF}(p_{\sigma_p,\varsigma}) = \bigcup_{\varsigma\in\Xi_{\bar{v}}} \mathsf{NF}(q_{\sigma_q,\varsigma})$. Due to symmetry it suffices to prove only that $t \in \mathsf{NF}(p_{\sigma_p,\varsigma})$ implies $t \in \bigcup_{\varsigma\in\Xi_{\bar{v}}} \mathsf{NF}(q_{\sigma_q,\varsigma})$ for every $\bar{v}$ and $\varsigma \in \Xi_{\bar{v}}$. Let t be an arbitrary element of $\mathsf{NF}(p_{\sigma_p,\varsigma})$. By Corollary 3.39 applied to (4.43) we have $\bigcup_{\bar{v}} \bigcup_{\varsigma\in\Xi_{\bar{v}}} \mathsf{NF}(p_{\sigma_p,\varsigma}) = \bigcup_{\bar{v}} \bigcup_{\varsigma\in\Xi_{\bar{v}}} \mathsf{NF}(q_{\sigma_q,\varsigma})$. By the assumption, t must belong to the left-hand side of this equation. Therefore, it belongs (up to α-equivalence) to the right-hand side, i.e. there exists $\bar{v}'$ such that $t \in \bigcup_{\varsigma\in\Xi_{\bar{v}'}} \mathsf{NF}(q_{\sigma_q,\varsigma})$. Note that $\bigcup_{\varsigma\in\Xi_{\bar{v}'}} \mathsf{NF}(q_{\sigma_q,\varsigma})$ consists of tight deterministic $\bar{v}'$-flat programs. Since t is tight deterministic $\bar{v}$-flat, by Lemma 4.20 $\bar{v} \doteq \bar{v}'$ and thus we have completed the proof of (4.44).

By Lemma 4.23 for every $\bar{v}$, (4.44) implies

$$\vdash_{\mathsf{ME}^\star} \sum_{\varsigma\in\Xi_{\bar{v}}} p_{\sigma_p,\varsigma} = \sum_{\varsigma\in\Xi_{\bar{v}}} q_{\sigma_q,\varsigma}. \tag{4.45}$$

Therefore, we can complete the 'equivalence' part of the lemma by the calculation: $\vdash_{\mathsf{ME}^\star} p\sigma = \sum_{\bar{v}} \sum_{\varsigma\in\Xi_{\bar{v}}} p_{\sigma_p,\varsigma} = \sum_{\bar{v}} \sum_{\varsigma\in\Xi_{\bar{v}}} q_{\sigma_q,\varsigma} = q\sigma$.

Let us prove the 'decidability' part. As we have seen previously,

$$\vdash_{\mathsf{ME}^\omega} p\sigma = q\sigma \implies \forall \bar v.\ \vdash_{\mathsf{ME}^\omega} \textstyle\sum_{\varsigma\in\Xi_{\bar v}} p_{\sigma_p,\varsigma} = \textstyle\sum_{\varsigma\in\Xi_{\bar v}} q_{\sigma_q,\varsigma} \implies \vdash_{\mathsf{ME}^\omega} p\sigma = q\sigma.$$

Therefore the problem of verifying $\vdash_{\mathsf{ME}^\omega} p\sigma = q\sigma$ (or $\vdash_{\mathsf{ME}^\star} p\sigma = q\sigma$) is equivalent to the problem of verifying $\vdash_{\mathsf{ME}^\omega} \sum_{\varsigma\in\Xi_{\bar v}} p_{\sigma_p,\varsigma} = \sum_{\varsigma\in\Xi_{\bar v}} q_{\sigma_q,\varsigma}$ for all $\bar v$, and thus we are done by Lemma 4.23. □

4.4 Weakly-iterative programs as a decidability candidate

The goal of this section is to lay the groundwork for extending the completeness and decidability result yet further. Specifically, we would like to re-establish it for the class of programs that we call weakly-iterative.

Definition 4.32 (Weak iterativity)**.** A program is *weakly-iterative* if every subterm of it having the form $(\mathsf{init}\, x \leftarrow p \,\mathsf{in}\, q^\star)$ is almost ret-free. In particular, every almost ret-free program is weakly-iterative.

The idea of the proof is again to provide suitable reductions, to bring a weakly-iterative program to a simpler form and then to call the completeness and decidability result established previously (Lemma 4.31). To that end, we need to develop the theory of flat programs a little further.

Note that one of the inconveniences about the notion of the flat program is that it is not stable under substitutions. E.g. given a flat program $(\mathsf{do}\ \langle x,y\rangle \leftarrow \mathsf{ret}\langle x,y\rangle; \mathsf{ret}\, x)$, already replacing y by a new variable z would produce $(\mathsf{do}\ \langle x,y\rangle \leftarrow \mathsf{ret}\langle x,z\rangle; \mathsf{ret}\, x)$ which does not match the definition of flatness.

Definition 4.33 (Stability)**.** Given a class of programs $\mathcal{C}$ and a set of variables V, we say that a program p is *V-stably in* $\mathcal{C}$ if for every vector of distinct variables $\bar v$ such that $\mathrm{Vars}(\bar v) \subseteq V$ and every appropriately typed vector of atomic programs $\bar a$, $p[\bar a/\bar v] \in \mathcal{C}$. In particular, if p is V-stably in $\mathcal{C}$, then $p \in \mathcal{C}$. We say that p is *stably in* $\mathcal{C}$ if it is $\mathrm{Vars}(p)$-stably in $\mathcal{C}$. Also, we shortcut '$\{v\}$-stable' to 'v-stable'. If $\mathcal{C}$ is the set of all flat programs, then we prefer to say '(V-)stably flat' rather than '(V-)stable in $\mathcal{C}$'.

Remark 4.34. Let $\mathcal{C}$ be a class of programs, closed under the operation of bulk replacement of a variable with a fresh variable. The targeted example here is of course the set of all flat programs. Given an arbitrary program that is stably in $\mathcal{C}$, we can always ensure by α-conversion that none of its bound variables come from any arbitrarily chosen set. Let e.g. p be stably in $\mathcal{C}$, and let V be the set of all variables (free and bound) occurring in p. Let V' be a set of fresh variables isomorphic to V, and, for every $z \in V$, let z' be the image of z under this isomorphism. For every vector of variables $\bar z$ such that $\mathrm{Vars}(\bar z) \subseteq V$ we denote by $\bar z'$ the vector obtained from $\bar z$ by replacing every z_i with

z'_i. Let p' be the program obtained by replacing every $z \in V$ by z' throughout. Then obviously p' is stably in $\mathcal{C}$. Since we might have renamed some free variables, p' is not necessarily α-equivalent to p (unless p is closed), but $p'[\bar{v}/\bar{v}']$ with $\bar{v}$ being such that $\mathrm{Vars}(\bar{v}) = \mathrm{Vars}(p)$ is α-equivalent to p. Since p' is stably in $\mathcal{C}$, $p'[\bar{v}/\bar{v}']$ is obviously also stably in $\mathcal{C}$.

In particular, given a stably $\bar{x}$-flat program p and a set of variables V, we can always switch to a stably $\bar{x}'$-flat program q, α-equivalent to p and such that $\mathrm{Vars}(\bar{x}') \cap V = \mathrm{Vars}(\bar{x}') \cap \mathrm{Vars}(q) = \emptyset$.

Lemma 4.35. *Let V be a set of variables and let $v \notin V$ be some variable. We put $V' := V \cup \{v\}$. Suppose we are given a V-stably $\bar{z}$-flat program p. Let $\bar{v}$ be a vector of distinct variables such that $\mathrm{Vars}(\bar{v}) = \mathrm{Vars}(\bar{z})\backslash\{v\}$. Then there exists an effectively computable pair of V'-stably flat programs p_1, p_2 such that*

$$\vdash_{\mathsf{ME}^\star} p = p_1 + \mathrm{do}\ \bar{v} \leftarrow p_2; \mathrm{ret}\, \bar{z}, \tag{4.46}$$

p_1 is $\bar{z}$-flat, p_2 is $\bar{v}$-flat and $p_2 \doteq \varnothing$ whenever $v \notin \mathrm{Vars}(\bar{z})$.

Proof. Note that we do not need to adhere the condition: $p_2 \doteq \varnothing$ whenever $v \notin \mathrm{Vars}(\bar{z})$ because it can always be ensured by redefining p_1 and p_2: $p_1 := p_1 + p_2$, $p_2 := \varnothing$. We construct p_1 and p_2 by induction over term complexity of p with respect to the clauses of Definition 4.13. To that end, we ensure first by Remark 4.21 that Kleene star occurs in p only in the form $(\mathrm{init}\, \bar{x} \leftarrow \mathrm{ret}\, \bar{x}\, \mathrm{in}\, q^\star)$. It can easily be seen that V-stability is maintained under the reduction rule of Remark 4.21.

FL1. We define p_1 and p_2 according to the specific form of p as follows.

If $p \doteq a_{i \triangleright \bar{z}}$ and $z_i = v$, then $p_1 := p$ and $p_2 := \varnothing$.

If $p \doteq a_{i \triangleright \bar{z}}$ with $z_i \neq v$, then $p_1 := \varnothing$ and $p_2 := a_{i \triangleright \bar{v}}$.

If $p \doteq \mathrm{ret}\, \bar{z}$, then $p_1 := \varnothing$ and $p_2 := \mathrm{ret}\, \bar{v}$.

If $p \doteq \varnothing$, then $p_1 := \varnothing$ and $p_2 := \varnothing$.

In all these cases the induction invariant is easily verified.

FL2. Let $p \doteq q + r$. Then, by induction, there exist q_1, q_2 satisfying the induction invariant for q and r_1, r_2 satisfying the induction for r. It is trivially verified that $p_1 := q_1 + r_1$ and $p_2 := q_2 + r_2$ satisfy the induction invariant for p.

FL3. Let $p \doteq (\mathrm{do}\ \bar{x} \leftarrow q; r)$. Then, by induction, there exist q_1, q_2 satisfying the induction invariant for q and r_1, r_2 satisfying the induction invariant for r. In particular,

$$\vdash_{\mathsf{ME}^\star} q = q_1 + \mathrm{do}\ \bar{y} \leftarrow q_2; \mathrm{ret}\, \bar{x},$$
$$\vdash_{\mathsf{ME}^\star} r = r_1 + \mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\, \bar{z}.$$

where $\mathrm{Vars}(\bar{y}) = \mathrm{Vars}(\bar{x})\backslash\{v\}$. Since p is V-stably flat, q must be V-stably flat and r must be $(V\backslash\mathrm{Vars}(\bar{x}))$-stably flat. Therefore, by induction hypothesis, q_1, q_2 are V'-stably flat, and r_1, r_2 are V''-stably flat with $V'' := (V\backslash\mathrm{Vars}(\bar{x})) \cup \{v\}$. Let us define:

$$\begin{aligned} p_1 &:= \mathrm{do}\ \bar{x} \leftarrow q_1; r_1 + \mathrm{do}\ \bar{y} \leftarrow q_2; r_1, \\ p_2 &:= \mathrm{do}\ \bar{x} \leftarrow q_1; r_2 + \mathrm{do}\ \bar{y} \leftarrow q_2; r_2. \end{aligned}$$

By definition, p_1 is $\bar{z}$-flat and p_2 is $\bar{v}$-flat. The proof of (4.46) runs as follows:

$$\begin{aligned} p &= \mathrm{do}\ \bar{x} \leftarrow (q_1 + \mathrm{do}\ \bar{y} \leftarrow q_2; \mathrm{ret}\,\bar{x}); r && \text{[by ind.]} \\ &= \mathrm{do}\ \bar{x} \leftarrow q_1; r + \mathrm{do}\ \bar{x} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow q_2; \mathrm{ret}\,\bar{x}); r && \text{[by } \mathbf{(dist_1^+)}\text{]} \\ &= \mathrm{do}\ \bar{x} \leftarrow q_1; r + \mathrm{do}\ \bar{y} \leftarrow q_2; \bar{x} \leftarrow \mathrm{ret}\,\bar{x}; r && \text{[by Lem. 4.1]} \\ &= \mathrm{do}\ \bar{x} \leftarrow q_1; r + \mathrm{do}\ \bar{y} \leftarrow q_2; r && \text{[by } \mathbf{(unit_2)}\text{]} \\ &= \mathrm{do}\ \bar{x} \leftarrow q_1; (r_1 + \mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}) + \\ &\quad\ \mathrm{do}\ \bar{y} \leftarrow q_2; (r_1 + \mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}) && \text{[by ind.]} \\ &= \mathrm{do}\ \bar{x} \leftarrow q_1; r_1 + \mathrm{do}\ \bar{x} \leftarrow q_1; (\mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}) + \\ &\quad\ \mathrm{do}\ \bar{y} \leftarrow q_2; r_1 + \mathrm{do}\ \bar{y} \leftarrow q_2; (\mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}) && \text{[by } \mathbf{(dist_2^+)}\text{]} \\ &= p_1 + \mathrm{do}\ \bar{x} \leftarrow q_1; (\mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}) + \\ &\quad\ \mathrm{do}\ \bar{y} \leftarrow q_2; (\mathrm{do}\ \bar{v} \leftarrow r_2; \mathrm{ret}\,\bar{z}). && \text{[by def. of } p_1\text{]} \end{aligned}$$

If $v \notin \mathrm{Vars}(\bar{z})$, then by induction $r_2 \doteq \varnothing$; the latter program evaluates to p_1, and thus we are done. Otherwise we proceed with the calculation

$$\begin{aligned} & p_1 + \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q_1; r_2); \mathrm{ret}\,\bar{z} + \\ & \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow q_2; r_2); \mathrm{ret}\,\bar{z} && \text{[by Lem. 4.1]} \\ &= p_1 + \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{x} \leftarrow q_1; r_2 + \mathrm{do}\ \bar{y} \leftarrow q_2; r_2); \mathrm{ret}\,\bar{z} && \text{[by } \mathbf{(dist_2^+)}\text{]} \\ &= p_1 + \mathrm{do}\ \bar{v} \leftarrow p_2; \mathrm{ret}\,\bar{z}. && \text{[by def. of } p_2\text{]} \end{aligned}$$

Let us prove that p_1 is V'-stably flat, i.e. p_1 is $\bar{x}$-stably flat for every $x \in V'$. By induction, both q_1 and q_2 are x-stably flat. If $x \notin \mathrm{Vars}(\bar{x})$ or $x = v$, then, by induction, both r_1 and r_2 are x-stably flat, and we are done. Otherwise, $x \in \mathrm{Vars}(\bar{x})\backslash\{v\} = \mathrm{Vars}(\bar{y})$ and therefore x may not occur freely in the subterms r_1, r_2, i.e. p_1 again must be x-free. The case of p_2 is proved analogously.

FL4. Let $p \doteq (\mathrm{do}\ \bar{z} \leftarrow \mathrm{ret}\,\bar{z}; q)$. Since by assumption p is V-stably flat, $\mathrm{Vars}(\bar{z}) \cap V = \emptyset$ and q is V-stably flat. By induction there exist V'-stably flat $\bar{z}$-flat q_1 and V'-stably flat $\bar{v}$-flat q_2 such that

$$\vdash_{\mathrm{ME}^\star} q = q_1 + \mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z}.$$

If $v \notin \mathrm{Vars}(\bar{z})$, then $q_2 \doteq \varnothing$ and we put $p_1 := (\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}\,q_1^\star)$, $p_2 := \varnothing$. So defined, p_1, p_2 trivially satisfy the induction invariant. In the case of $v \in \mathrm{Vars}(\bar{z})$ let us define:

$$\begin{aligned} p_1 &:= \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,q_2^\star\,\mathrm{res}\,\bar{v} \to (\mathrm{init}\,\bar{z} \leftarrow q_1\,\mathrm{in}\,q^\star) \\ p_2 &:= \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,q_2^\star. \end{aligned}$$

It is easily verified that p_1 is $\bar{z}$-flat and p_2 is $\bar{v}$-flat. By Lemma 4.1, we have:

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{z} \leftarrow \mathrm{ret}\,\bar{z}; (\mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z}) = \mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z}$$

and therefore by Corollary 4.8, the identity

$$\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z})^\star = \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,q_2^\star\,\mathrm{res}\,\bar{v} \to \mathrm{ret}\,\bar{z} \tag{4.47}$$

must provable in $\mathsf{ME}^\star$. Now the proof of (4.46) runs as follows:

$$\begin{aligned} p = {} & \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(q_1 + \mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z})^\star \\ = {} & \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z})^\star + \\ & \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow q_2; \mathrm{ret}\,\bar{z})^\star \\ & \mathrm{res}\,\bar{z} \to (\mathrm{init}\,\bar{z} \leftarrow q_1\,\mathrm{in}\,q^\star) && \text{[by Lem. 4.4]} \\ = {} & \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,q_2^\star\,\mathrm{res}\,\bar{v} \to \mathrm{ret}\,\bar{z} + \\ & \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,q_2^\star\,\mathrm{res}\,\bar{v} \to (\mathrm{init}\,\bar{z} \leftarrow q_1\,\mathrm{in}\,q^\star) && \text{[by 4.47]} \\ = {} & p_1 + p_2. \end{aligned}$$

Finally, prove that both p_1 and p_2 are indeed V'-stably flat, i.e. p_1, p_2 are x-stably flat for every $x \in V' = V \cup \{v\}$. If $x \neq v$, then $x \notin \mathrm{Vars}(\bar{z})$ because, as it was argued above, $\mathrm{Vars}(\bar{z}) \cap V = \varnothing$. Then we are immediately done by the induction hypothesis. Let $x = v$. Recall that $v \notin \mathrm{Vars}(\bar{v})$. By induction hypothesis, v may occur freely neither in q_1 nor in q_2. Possible occurrences of v in p_1 under q are bound. In summary, v may occur freely neither in p_1 nor in p_2, i.e. both p_1, p_2 are v-stably flat. □

Lemma 4.36. *Let $\bar{z}$ be a vector of distinct variables. We denote by Z the set of all vectors of distinct variables $\bar{z}^k$ such that* $\mathrm{Vars}(\bar{z}^k) \subseteq \mathrm{Vars}(\bar{z})$. *Given a $\bar{z}$-flat program p, there exists an effectively found sequence of programs $\bar{p}$, every element p_k of which is stably flat, $\bar{z}^k$-flat and*

$$\vdash_{\mathsf{ME}^\star} p = \textstyle\sum_k \mathrm{do}\ \bar{z}^k \leftarrow p_k; \mathrm{ret}\,\bar{z}. \tag{4.48}$$

Proof. Let V be an arbitrary subset of $\mathrm{Vars}(p)$. We prove the claim by induction over $n := |V|$, but replace the word 'stability' with 'V-stability'. This will imply the original claim under the assignment: $V := \mathrm{Vars}(p)$.

If $n = 0$, then p is stably flat. We immediately have a solution: $\bar{p} := (\mathsf{do}\ \star \leftarrow \mathsf{ret}\,\star; p)$.

Suppose, $n > 0$. Let v be some variable from V, and let $V' := V \backslash \{v\}$. By induction hypothesis, there exists a vector of V'-stably flat programs $\bar{q}$ such that every q_k is $\bar{z}^k$-flat and $\vdash_{\mathsf{ME}^\star} p = \sum_i \mathsf{do}\ \bar{z}^k \leftarrow q_k; \mathsf{ret}\,\bar{z}$. By Lemma 4.35, for every i there exist V-stably flat $\bar{z}^k$-flat u_k and V-stably flat $\bar{v}^k$-flat r_k such that $\vdash_{\mathsf{ME}^\star} q_k = u_k + \mathsf{do}\ \bar{v}^k \leftarrow r_k; \mathsf{ret}\,\bar{z}^k$ where $\bar{v}^k$ satisfies the equality $\mathrm{Vars}(\bar{v}^k) = \mathrm{Vars}(\bar{z}^k) \backslash \{v\}$. Then

$$\begin{aligned} p &= \textstyle\sum_i \mathsf{do}\ \bar{z}^k \leftarrow q_k; \mathsf{ret}\,\bar{z} \\ &= \textstyle\sum_i \mathsf{do}\ \bar{z}^k \leftarrow (u_k + \mathsf{do}\ \bar{v}^k \leftarrow r_k; \mathsf{ret}\,\bar{z}^k); \mathsf{ret}\,\bar{z} \\ &= \textstyle\sum_i \mathsf{do}\ \bar{z}^k \leftarrow u_k; \mathsf{ret}\,\bar{z} + \sum_i \mathsf{do}\ \bar{z}^k \leftarrow (\mathsf{do}\ \bar{v}^k \leftarrow r_k; \mathsf{ret}\,\bar{z}^k); \mathsf{ret}\,\bar{z} && [\text{by } (\mathbf{dist}_1^+)] \\ &= \textstyle\sum_i \mathsf{do}\ \bar{z}^k \leftarrow u_k; \mathsf{ret}\,\bar{z} + \sum_i \mathsf{do}\ \bar{v}^k \leftarrow r_k; \mathsf{ret}\,\bar{z}. && [\text{by Lem. } 4.1] \end{aligned}$$

Note that for every k, $\bar{v}^k \in Z$. Therefore we can regroup the latter sum so as to obtain the presentation in question. □

Definition 4.37 (Layered programs)**.** We define *layered* programs by induction over the return type by the following clauses:

- every atomic program is layered;
- if p and q are layered, then $\langle p, q\rangle$ is also layered;
- every program of the form $\left(\sum_i \mathsf{do}\ \bar{x}^i \leftarrow p_i; \mathsf{ret}\,q_i\right)$ is layered, provided all the q_i are layered, and for every i, p_i is stably $\bar{x}^i$-flat.

Similar to the case of flat programs, we refer to the programs which are V-stably in the class of layered programs as *V-stably layered*.

Lemma 4.38. *Given a layered program p, there is an effectively computable stably layered program q such that $\vdash_{\mathsf{ME}^\star} p = q$.*

Proof. We prove the claim by induction over the return type of p. Let us proceed by case distinction over the clauses of Definition 4.37. If p is atomic, evidently it is already stably layered, and thus we can define $q := p$. Let $p \doteq \langle r, u\rangle$, and let r' and u' be layered programs provably equal to r and u correspondingly whose existence is guaranteed by the induction hypothesis. Then the assignment $q := \langle r', u'\rangle$ evidently provides a correct value for q. Finally, suppose $p \doteq \left(\sum_i \mathsf{do}\ \bar{x}^i \leftarrow u_i; r_i\right)$. Since all the r_i must be layered, by induction hypothesis, for every i there exists stably layered r_i', provably equal to r_i. By Lemma 4.36, for every i there exists a vector of programs $\bar{u}^i$ such that for every j, u^i_j is stably $\bar{x}^{i,j}$-flat, $\mathrm{Vars}(\bar{x}^{i,j}) \subseteq \mathrm{Vars}(\bar{x}^i)$ and $\vdash_{\mathsf{ME}^\star} u_i = \sum_j \mathsf{do}\ \bar{x}^{i,j} \leftarrow u^i_j; \mathsf{ret}\,\bar{x}^i$. By evident calculations the equation

$$p = \textstyle\sum_{i,j} \mathsf{do}\ \bar{x}^{i,j} \leftarrow u^i_j; \mathsf{ret}\, r_i$$

must be provable in $\mathsf{ME}^\star$. Observe that the right-hand side of it is stably layered. Therefore we can take it as a value for q. □

Lemma 4.39. *Let p be a layered program, and let q be a flat program. Then* $(\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q)$ *is provably equivalent under* $\mathsf{ME}^\star$ *to an effectively computable layered program.*

Proof. By Lemma 4.28, let us assume w.l.o.g. that q is tight. By Remark 4.21, we ensure that Kleene star occurs in q only in the form $(\mathrm{init}\, \bar z \leftarrow \mathrm{ret}\, \bar z\, \mathrm{in}\, u^\star)$.

Let n be the length of the longest footprint among all the flat subterms of q and p. Let V be a finite set of variables containing all the variables (free and bound) occurring in q and p plus $3n$ fresh variables of every type A for which there exists a subterm of q whose return type is A. Let R be the set of programs of the form $t\sigma$ where t ranges over layered subterms of p and σ ranges over all injective substitutions $\mathrm{Vars}(t) \to V$. Evidently, R is finite and $p \in R$. By Lemma 4.38, all the elements of R are layered.

Let $\bar v$ be a vector of distinct variables such that $\mathrm{Vars}(\bar v) = V$. Let us generalise the claim of the lemma as follows. We prove that for every vector $\bar t$ such that all the t_j are from $R \cup V$, there exist effectively computable $\bar v$-flat q_i and $t^i_j \in R \cup V$ such that

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar x \leftarrow \mathrm{ret}\, \bar t; q = \textstyle\sum_i \mathrm{do}\ \bar v \leftarrow q_i; \mathrm{ret}\, \bar t^i. \tag{4.49}$$

Let us show how (4.49) implies the original claim. Since $p \in R \cup V$, we can take $\bar t := p$ in (4.49) and thus we are left to show that the right-hand side of (4.49) can be transformed to a layered program. By Lemma 4.36, for every q_i there exists an effectively computable vector of programs $\bar q^i$ such that $\vdash_{\mathsf{ME}^\star} q_i = \sum_k \mathrm{do}\ \bar v^k \leftarrow q^i_k; \mathrm{ret}\, \bar v$, and for every k, q^i_k is stably $\bar v^k$-flat and $\mathrm{Vars}(\bar v^k) \subseteq \mathrm{Vars}(\bar v)$. By Lemma 4.1, the right-hand side of (4.49) is provably equal to $\left(\sum_{i,k} \mathrm{do}\ \bar v^k \leftarrow q^i_k; \mathrm{ret}\, \bar t^i\right)$ which is layered.

We proceed with the proof of (4.49). First of all, let ensure that $\mathrm{Vars}(\bar x) = \mathrm{Vars}(q)$ by cutting from $\bar t$ and $\bar x$ the redundant components and completing both $\bar x$ and $\bar t$ with the missing variables from $\mathrm{Vars}(q)\backslash\mathrm{Vars}(\bar x)$, if any. Let us proceed by induction over the term complexity of q with respect to the clauses of Definition 4.13.

FL1. If $q \doteq \varnothing$, then we are done by taking the sum of zero elements on the right-hand side of (4.49).

If $q \doteq \mathrm{ret}\, \bar z$, then we put $\bar q := \mathrm{ret}\, \bar v$ and $\bar p^1 := \mathrm{ret}\, \bar z[\bar t/\bar x]$. It is clear that these programs indeed come from the declared program classes. The proof of (4.49) is simple: $\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar x \leftarrow \mathrm{ret}\, \bar t; q = \mathrm{ret}\, \bar z[\bar t/\bar x] = \mathrm{do}\ \bar v \leftarrow \mathrm{ret}\, \bar v; \mathrm{ret}\, \bar z[\bar t/\bar x]$.

Let $q \doteq a_{k \triangleright \bar z}$. Let z_0 be a fresh variable whose type is the same as the return type of a. Now $a_{k \triangleright \bar z}$ is provably equal to $(\mathrm{do}\ z_k \leftarrow a; \mathrm{ret}\, \bar z)$, no matter if $k = 0$ or not. Let σ be the restriction of the substitution $[\bar t/\bar x]$ to the set of those x_j which are distinct from z_k. In particular, if $k = 0$, σ coincides with $[\bar{t}/\bar{x}]$. Note that the left-hand side of (4.49) is provably equal to $(\mathrm{do}\ z_k \leftarrow a[\bar{t}/\bar{x}]; \mathrm{ret}\,\bar{z}\sigma)$.

W.l.o.g. we assume that a is normal. Then it must be in either of the forms: $h_1(\ldots(h_m(v))\ldots)$ or $h_1(\ldots(h_m(f(w)))\ldots)$, where $f \in \Sigma$, v is a variable, and for all l, $h_l \in \{\mathrm{fst}, \mathrm{snd}\}$. Consider these two subcases individually.

– Let $a \doteq h_1(\ldots(h_m(v))\ldots)$. Assume that v is distinct from any of the x_j. Then $a[\bar{t}/\bar{x}] \doteq a$ and thus the left-hand side of (4.49) is provably equal to $(\mathrm{do}\ z_k \leftarrow a; \mathrm{ret}\,\bar{z}\sigma)$. Let us put $\bar{q} := a_{l \triangleright \bar{v}}$ and $\bar{p}^1 := \mathrm{ret}\,\bar{z}\sigma$, where l is the number of z_k in $\bar{v}$ if $k > 0$ and 0 otherwise. We have thus:

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{x} \leftarrow \mathrm{ret}\,\bar{t}; q = \mathrm{do}\ z_k \leftarrow a; \mathrm{ret}\,\bar{z}\sigma = \mathrm{do}\ \bar{v} \leftarrow a_{l \triangleright \bar{v}}; \mathrm{ret}\,\bar{z}\sigma$$

which proves (4.49). The remainder of the claim is clearly satisfied.

If for some j, $v \doteq x_j$, then $a[\bar{t}/\bar{x}] \doteq h_1(\ldots(h_m(t_j))\ldots)$. By the definition of layered programs, and for typing reasons, the latter program must be provably equal to a program of the form $(\sum_i \mathrm{do}\ \bar{y}^i \leftarrow s_i; \mathrm{ret}\,r_i)$ where all the r_i are layered subterms of p_j and for every i, s_i is $\bar{y}^i$-flat. The left-hand side of (4.49) is then provably equal to

$$\textstyle\sum_i \mathrm{do}\ z_k \leftarrow (\mathrm{do}\ \bar{y}^i \leftarrow s_i; \mathrm{ret}\,r_i); \mathrm{ret}\,\bar{z}\sigma. \tag{4.50}$$

Let us argue that for every i, there exists a vector $\bar{v}^i$ of variables from V such that $\bar{v}^i$ has the same return type as $\bar{y}^i$ and

$$\mathrm{Vars}(\bar{v}^i) \cap \mathrm{Vars}(\bar{z}\sigma) = \mathrm{Vars}(\bar{v}^i) \cap \mathrm{Vars}(r_i) = \emptyset\,. \tag{4.51}$$

Since $|\bar{z}\sigma| = |\bar{z}| \leqslant n$ and $|\mathrm{Vars}(r_i)| \leqslant n$, $|\mathrm{Vars}(\bar{z}\sigma) \cup \mathrm{Vars}(r_i)| \leqslant 2n$. Therefore for every j, $V \setminus (\mathrm{Vars}(\bar{z}\sigma) \cup \mathrm{Vars}(r_i))$ must contain at least n distinct variables of the same type as y^i_j. Hence we can find a vector of variables $\bar{v}^i$ of appropriate types entirely consisting of elements of the latter set. This vector will evidently satisfy (4.51). We can now transform (4.50) as follows:

$$\begin{aligned}
&\textstyle\sum_i \mathrm{do}\ z_k \leftarrow (\mathrm{do}\ \bar{y}^i \leftarrow s_i; \mathrm{ret}\,r_i); \mathrm{ret}\,\bar{z}\sigma \\
&\quad = \textstyle\sum_i \mathrm{do}\ z_k \leftarrow (\mathrm{do}\ \bar{v}^i \leftarrow s_i; \mathrm{ret}\,r_i[\bar{v}^i/\bar{y}^i]); \mathrm{ret}\,\bar{z}\sigma \\
&\quad = \textstyle\sum_i \mathrm{do}\ \bar{v}^i \leftarrow s_i; z_k \leftarrow \mathrm{ret}\,r_i[\bar{y}^i/\bar{v}^i]; \mathrm{ret}\,\bar{z}\sigma && \text{[by Lem. 4.1]} \\
&\quad = \textstyle\sum_i \mathrm{do}\ \bar{v}^i \leftarrow s_i; \mathrm{ret}\,\bar{z}\sigma\big[r_i[\bar{y}^i/\bar{v}^i]/z_k\big]. && \text{[by } \textbf{(unit}_2\textbf{)}\text{]} \\
&\quad = \textstyle\sum_i \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{v}^i \leftarrow s_i; \mathrm{ret}\,\bar{v}); \\
&\qquad\quad \mathrm{ret}\,\bar{z}\sigma\big[r_i[\bar{y}^i/\bar{v}^i]/z_k\big]. && \text{[by Lem. 4.1]}
\end{aligned}$$

Now we can complete the proof of the induction invariant by putting for every i: $q_i := (\mathsf{do}\ \bar{v}^i \leftarrow s_i; \mathsf{ret}\,\bar{v})$ and $\bar{p}^i := \bar{z}\sigma\big[r_i[\bar{y}^i/\bar{v}^i]/z_k\big]$. The only non-trivial part of it to be shown is that every p^i_j is an element of $R \cup V$. If $j \neq k$, then $p^i_j \doteq z_j\sigma$ is evidently an element of $R \cup V$. If $j = k$, then $p^i_j \doteq r_i[\bar{y}^i/\bar{v}^i]$ is also an element of $R \cup V$, because by assumption $r_i \in R \cup V$ and by (4.51), $[\bar{y}^i/\bar{v}^i]$ can be trivially completed to an injective substitution over all $\mathrm{Vars}(r_i)$.

– Let $a \doteq h_1(\ldots(h_m(f(w)))\ldots)$. By Lemma 3.22, $a[\bar{t}/\bar{x}]$ must be atomic. Let us denote it by b and let us put $\bar{q} := b_{l \triangleright \bar{v}}$ and $\bar{p}^1 := \mathsf{ret}\,\bar{z}\sigma$, where l is the index of z_k in $\bar{v}$ if $k > 0$ and 0 otherwise. The proof of (4.49) is as follows:

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{t}; q = \mathsf{do}\ z_k \leftarrow b; \mathsf{ret}\,\bar{z}\sigma = \mathsf{do}\ \bar{v} \leftarrow b_{l \triangleright \bar{v}}; \mathsf{ret}\,\bar{z}\sigma.$$

The remaining conditions that should be satisfied by the q_i and the p^i_j are easily verified.

FL2. If $q \doteq (u + r)$, then the goal trivially follows by the induction hypothesis: we only need to merge the sums of the presentation (4.49) for q and for r.

FL3. Suppose $q \doteq (\mathsf{do}\ \bar{y} \leftarrow u; r)$. Let us transform the left-hand side of (4.49) equivalently under $\mathsf{ME}^\star$ calling by need the induction hypothesis:

$$\begin{aligned}
&\mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{t}; q \\
&\quad\doteq \mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{t}; \bar{y} \leftarrow u; r \\
&\quad= \mathsf{do}\ \bar{y} \leftarrow (\mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{t}; u); r && \text{[by Lem. 4.1]} \\
&\quad= \mathsf{do}\ \bar{y} \leftarrow \Big(\textstyle\sum_j \mathsf{do}\ \bar{v} \leftarrow u_j; \mathsf{ret}\,\bar{s}^j\Big); r && \text{[by ind.]} \\
&\quad= \textstyle\sum_j \mathsf{do}\ \bar{v} \leftarrow u_j; \bar{y} \leftarrow \mathsf{ret}\,\bar{s}^j; r && \text{[by Lem. 4.1]} \\
&\quad= \textstyle\sum_j \mathsf{do}\ \bar{v} \leftarrow u_j; \Big(\textstyle\sum_k \mathsf{do}\ \bar{v} \leftarrow r_k; \mathsf{ret}\,\bar{t}^k\Big) && \text{[by ind.]} \\
&\quad= \textstyle\sum_{j,k} \mathsf{do}\ \bar{v} \leftarrow (\mathsf{do}\ \bar{v} \leftarrow u_j; r_k); \mathsf{ret}\,\bar{t}^k. && \text{[by Lem. 4.1]}
\end{aligned}$$

Here, the properties ensured by the induction hypothesis are as follows: for every j, u_j is $\bar{v}$-flat; for every k, r_k is $\bar{v}$-flat; and all the elements of the $\bar{s}^j$ and the $\bar{t}^k$ belong to $R \cup V$. It is evident that the last program in the latter calculation matches the right-hand side of (4.49). The precise assignments for the programs in question can be derived easily.

FL4. Let $q \doteq (\mathsf{init}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{x}\ \mathsf{in}\ u^\star)$. The set of all vectors of programs from $R \cup V$ having the same type as $\bar{x}$ is evidently finite. Let n be the number of all such vectors, and let $\bar{s}^k$ denote the k-th one. We agree that $\bar{t} \doteq \bar{s}^1$. By induction hypothesis, for every m,

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{s}^m; q = \textstyle\sum_k \mathsf{do}\ \bar{v} \leftarrow r_{m,k}; \mathsf{ret}\,\bar{s}^k$$

for some effectively computable vector of $\bar{v}$-flat programs $\bar{r}$. It can easily be seen that $(n, \lambda k.\,\bar{v}, \lambda m.\,\lambda k.\,r_{m,k})$ makes a computational net. Let $(n, \lambda k.\,\bar{v}, \lambda m.\,\lambda k.\,\hat{r}_{m,k})$

be the transitive closure of it. By Lemma 4.6, whose side conditions follow trivially,

$$\vdash_{\mathsf{ME}^\star} \mathsf{do}\ \bar{x} \leftarrow \mathsf{ret}\,\bar{t}; q = \textstyle\sum_k \mathsf{do}\ \bar{v} \leftarrow \hat{r}_{1,k}; \mathsf{ret}\,\bar{s}^k.$$

By induction over the definition of transitive closure, we can easily conclude that every $\hat{r}_{m,k}$ is $\bar{v}$-flat. The presentation (4.49) now becomes apparent. □

Lemma 4.40. *Let p be an almost* ret*-free program such that*

$$\vdash_{\mathsf{ME}^\star} p = \textstyle\sum_i \mathsf{do}\ \bar{x}^i \leftarrow p_i; \mathsf{ret}\, q_i \tag{4.52}$$

where for every i, p_i is $\bar{x}^i$-flat and q_i is layered (i.e. the right-hand side of (4.52) is layered). Then for every i, either $\mathsf{tr}(p_i) \doteq \varnothing$ or $\mathsf{nf}(q_i)$ is cartesian.

Proof. By Corollary 3.35, for every i, $\mathsf{NF}(\mathsf{do}\ \bar{x}^i \leftarrow p_i; \mathsf{ret}\, q_i) \lesssim_\star \mathsf{NF}(p)$. By Lemma 4.11 and Corollary 4.12, for every i we have:

$$\{\mathsf{nf}(\mathsf{do}\ \bar{x}^i \leftarrow t; \mathsf{ret}\, q_i) \mid t \in \mathsf{tr}(p_i)\} \lesssim_\star \{\mathsf{nf}(s) \mid s \in \mathsf{tr}(p)\}. \tag{4.53}$$

It is easy to see that normalisation under $\rightarrowtail_{\mathsf{m}}$ respects almost ret-freeness. Therefore, for every $s \in \mathsf{tr}(p)$, $\mathsf{nf}(s)$ must be ret-free. By (4.53), this means that for every $t \in \mathsf{tr}(p_i)$, $\mathsf{nf}(\mathsf{do}\ \bar{x}^i \leftarrow t; \mathsf{ret}\, q_i)$ must be almost ret-free. If $\mathsf{tr}(p_i) \doteq \varnothing$, then we are done. Otherwise, let us fix some $t \in \mathsf{tr}(p_i)$. Since p_i is flat, t is deterministic. Therefore, $\mathsf{nf}(t)$ must have the form $(\mathsf{do}\ \bar{z} \leftarrow \bar{a}; u)$, where all the a_k are atomic and u is either atomic or administrative. Then

$$\begin{aligned}
&\mathsf{nf}(\mathsf{do}\ \bar{x}^i \leftarrow t; \mathsf{ret}\, q_i) \\
&\quad \doteq \mathsf{nf}(\mathsf{do}\ \bar{x}^i \leftarrow (\mathsf{do}\ \bar{z} \leftarrow \bar{a}; u); \mathsf{ret}\, \mathsf{nf}(q_i)) \\
&\quad \doteq \mathsf{do}\ \bar{z} \leftarrow \bar{a}; \mathsf{nf}(\mathsf{do}\ \bar{x}^i \leftarrow u; \mathsf{ret}\, \mathsf{nf}(q_i)).
\end{aligned}$$

It can easily be seen from the form of the reduction rules that the latter program is almost ret-free only if $\mathsf{nf}(q_i)$ is cartesian. □

Lemma 4.41. *Every weakly iterative program p can be effectively and soundly under* $\mathsf{ME}^\star$ *reduced to a layered program.*

Proof. W.l.o.g. we assume that p is normal. Then we proceed by induction over the term complexity of A, the return type of p. If A is T-free, then by Lemma 3.22, p must be atomic. Since

$$\vdash_{\mathsf{ME}^\star} p = \mathsf{do}\ x \leftarrow p; \mathsf{ret}\, x$$

we can return as the result $(\mathsf{do}\ x \leftarrow p; \mathsf{ret}\, x)$, which is evidently layered.

Let $A = B \times C$. By the induction hypothesis there exist effectively computable layered programs q, r, provably equal to $\mathrm{fst}(p)$ and $\mathrm{snd}(p)$ correspondingly. By Definition 4.37, $\langle q, r\rangle$ is layered. Obviously $\vdash_{\mathsf{ME}^\star} p = \langle u, r\rangle$, and thus we are done.

Finally, let $A = TB$ with some B. We proceed by further induction over the term complexity of p. W.l.o.g. p is normal. By Lemma 3.22, there are the following options for p: p is atomic, $p \doteq \varnothing$, $p \doteq (q + r)$, $p \doteq (\mathrm{do}\ x \leftarrow q; r)$, and $p \doteq (\mathrm{init}\, x \leftarrow q\, \mathrm{in}\, r^\star)$. Only the two later cases do not immediately follow from the inner induction hypothesis.

Suppose $p \doteq (\mathrm{do}\ x \leftarrow q; r)$. By the inner induction and by Lemma 4.38, q can be reduced to a stably layered program. If this layered program, say u, happens to be atomic, we transform it further under the first unit law to $(\mathrm{do}\ y \leftarrow u; \mathrm{ret}\, y)$. The same reasoning can be applied to r. In summary we obtain provable equalities:

$$\begin{aligned} \vdash_{\mathsf{ME}^\star} q &= \textstyle\sum_i \mathrm{do}\ \bar{x}^i \leftarrow q_i; \mathrm{ret}\, s_i, \\ \vdash_{\mathsf{ME}^\star} r &= \textstyle\sum_j \mathrm{do}\ \bar{y}^j \leftarrow r_j; \mathrm{ret}\, t_j \end{aligned}$$

where for every i, q_i is stably $\bar{x}^i$-flat; for every j, r_j is stably $\bar{y}^j$-flat; and all the s_i, r_j are stably layered. By Remark 4.34, we ensure that for every i, j, $\mathrm{Vars}(\bar{x}^i) \cap \mathrm{Vars}(\bar{y}^j) = \varnothing$, $x \notin \mathrm{Vars}(\bar{x}^i)$ and $x \notin \mathrm{Vars}(\bar{y}^j)$. Let us transform p under $\mathsf{ME}^\star$ as follows:

$$\begin{aligned} p &\doteq \mathrm{do}\ x \leftarrow q; r \\ &= \mathrm{do}\ x \leftarrow \Big(\textstyle\sum_i \mathrm{do}\ \bar{x}^i \leftarrow q_i; \mathrm{ret}\, s_i\Big); \Big(\textstyle\sum_j \mathrm{do}\ \bar{y}^j \leftarrow r_j; \mathrm{ret}\, t_j\Big) \\ &= \textstyle\sum_{i,j} \mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{y}^j \leftarrow (\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; r_j); \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; t_j). \end{aligned}$$

By Lemma 4.39, for every i, j there exists a vector of stably $\bar{z}^{i,j}$-flat programs $\bar{r}^{i,j}$ and a vector of layered programs $\bar{u}^{i,j}$ such that

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; r_j = \textstyle\sum_k \mathrm{do}\ \bar{z}^{i,j} \leftarrow r^{i,j}_k; \mathrm{ret}\, u^{i,j}_k. \tag{4.54}$$

By Remark 4.34, we ensure that the variables in the $\bar{z}^{i,j}$ are distinct from the variables introduced earlier. Now we continue the previous computation as follows:

$$\begin{aligned} &\textstyle\sum_{i,j} \mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{y}^j \leftarrow (\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; r_j); \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; t_j) \\ &\quad = \textstyle\sum_{i,j,k} \mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{y}^j \leftarrow \big(\mathrm{do}\ \bar{z}^{i,j} \leftarrow r^{i,j}_k; \mathrm{ret}\, u^{i,j}_k\big); \\ &\qquad\qquad \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; t_j) && \text{[by 4.54]} \\ &\quad = \textstyle\sum_{i,j,k} \mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{z}^{i,j} \leftarrow r^{i,j}_k; \bar{y}^j \leftarrow \mathrm{ret}\, u^{i,j}_k; \\ &\qquad\qquad \mathrm{ret}(\mathrm{do}\ x \leftarrow \mathrm{ret}\, s_i; t_j) && \text{[by Lem. 4.1]} \\ &\quad = \textstyle\sum_{i,j,k} \mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{z}^{i,j} \leftarrow r^{i,j}_k; \\ &\qquad\qquad \mathrm{ret}\big(\mathrm{do}\ \bar{y}^j \leftarrow \mathrm{ret}\, u^{i,j}_k; x \leftarrow \mathrm{ret}\, s_i; t_j\big) && \text{[by } \mathbf{(unit_2)}\text{]} \\ &\quad = \textstyle\sum_{i,j,k} \mathrm{do}\ \langle \bar{x}^i, \bar{z}^{i,j}\rangle \leftarrow (\mathrm{do}\ \bar{x}^i \leftarrow q_i; \bar{z}^{i,j} \leftarrow r^{i,j}_k; \mathrm{ret}\langle \bar{x}^i, \bar{z}^{i,j}\rangle); \end{aligned}$$

$$\mathrm{ret}\big(\mathrm{do}\ \bar{y}^j \leftarrow \mathrm{ret}\, u_k^{i,j}; x \leftarrow \mathrm{ret}\, s_i; t_j\big). \qquad \text{[by } \textbf{(unit}_\mathbf{2}\textbf{)}\text{]}$$

Observe that for every i, j, k the return type of $\big(\mathrm{do}\ \bar{y}^j \leftarrow \mathrm{ret}\, u_k^{i,j}; x \leftarrow \mathrm{ret}\, s_i; t_j\big)$ is B, i.e. simpler than A. Therefore the latter program can be replaced with a layered program by the outer induction hypothesis. According to the previous calculations, this means that the original program p can be reduced to a layered program, which proves the claim.

Finally, let us consider the remaining case: $p \doteq (\mathrm{init}\, x \leftarrow q\, \mathrm{in}\, r^\star)$. Since p is weakly iterative, by definition p and therefore q must be moreover almost ret-free. By virtue of the provable equality

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\, x \leftarrow q\, \mathrm{in}\, r^\star = q + \mathrm{do}\ x \leftarrow q; (\mathrm{init}\, x \leftarrow \mathrm{ret}\, x\, \mathrm{in}\, r^\star)$$

it suffices to prove the case when $q \doteq x$. The goal will then follow by the proved clauses for binding and choice. We assume that $q \doteq x$ henceforth. By the inner induction hypothesis and Lemma 4.38, we can prove in $\mathsf{ME}^\star$ the equality

$$r = \textstyle\sum_i \mathrm{do}\ \bar{x}^i \leftarrow r_i; \mathrm{ret}\, s_i$$

with some effectively computable stably layered program on the right-hand side. By Remark 4.34, we ensure that all the variable names in the $\bar{x}^i$ are new with respect to the ones introduced earlier. By Lemma 4.40, we assume henceforth w.l.o.g. that all the s_i are normal cartesian. Let $\bar{x}$ be a vector of distinct variables such that $\mathrm{Vars}(\bar{x}) = \bigcup_i \mathrm{Vars}(\bar{x}^i)$, and for every i, let $u_i := (\mathrm{do}\ \bar{x}^i \leftarrow r_i; \mathrm{ret}\, \bar{x})$. After elementary calculations we have:

$$\vdash_{\mathsf{ME}^\star} r = \textstyle\sum_i \mathrm{do}\ \bar{x} \leftarrow u_i; \mathrm{ret}\, s_i. \tag{4.55}$$

Let R be the set of all normal cartesian programs whose return type is B and whose free variables all come from $\mathrm{Vars}(\bar{x}) \cup \bigcup_i \mathrm{Vars}(r_i) \cup \mathrm{Vars}(x)$. Then by definition, for every i, $s_i \in R$. Let us argue that R is finite. Since all the elements of R are normal cartesian, each of them is a tree where branching is realised by the pairing operator, and every terminal node is either $\star$ or $h_1(\ldots(h_m(v))\ldots)$ with $v \in \mathrm{Vars}(\bar{x}) \cup \bigcup_i \mathrm{Vars}(r_i) \cup \mathrm{Vars}(x)$ and $h_k \in \{\mathrm{fst}, \mathrm{snd}\}$ for all k. There are only a finite number of trees of this kind, because the set of programs which can play the role of terminal nodes is finite, and every pairing produces a program whose return type is more complex then the return type of the arguments. We enumerate all the elements of R, denote the j-th one by t_j and agree that $t_1 = x$. Then we can convert (4.55) to

$$\vdash_{\mathsf{ME}^\star} r = \textstyle\sum_j \mathrm{do}\ \bar{x} \leftarrow u_j; \mathrm{ret}\, t_j$$

by completing the sum in (4.55) with components of the form $(\mathrm{do}\ \bar{x} \leftarrow \varnothing; \mathrm{ret}\, t_j)$ for every j such that t_j is not equal to any of the r_i. Observe that for every k:

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ x \leftarrow \mathrm{ret}\, t_k; r = \textstyle\sum_j \mathrm{do}\ \bar{x} \leftarrow u_j[t_k/x]; \mathrm{ret}\, \mathsf{nf}\big(t_j[t_k/x]\big).$$

Since by assumption all the u_j are stably $\bar{x}$-flat, all the $u_i[t_k/x]$ are also stably $\bar{x}$-flat. By definition, $\mathsf{nf}\big(t_j[t_k/x]\big) \in R$. Therefore we can regroup the components of the latter sum and switch to

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \ x \leftarrow \mathrm{ret}\, t_k; r = \textstyle\sum_j \mathrm{do}\ \ \bar{x} \leftarrow u_{k,j}; \mathrm{ret}\, t_j.$$

where for every k, j, $u_{k,j}$ is stably $\bar{x}$-flat. By Lemma 4.6, whose side conditions are trivially verified, we have:

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\, x \leftarrow \mathrm{ret}\, x\, \mathrm{in}\, r^\star = \textstyle\sum_j \mathrm{do}\ \ \bar{x} \leftarrow \hat{u}_{1,j}; \mathrm{ret}\, t_j. \tag{4.56}$$

for some family of $\bar{x}$-flat programs $\hat{u}_{1,j}$. The right-hand side of (4.56) is thus layered. The proof is now completed. □

4.5 Completeness and decidability, the ultimate

Throughout this section let $\mathcal{E}$ be an arbitrary data theory over the underlying signature Σ.

Definition 4.42 (Superflat programs)**.** We call a $\bar{z}$-flat program *$\bar{z}$-superflat* if all its flat subterms are $\bar{z}$-flat. We call a program superflat if it is $\bar{z}$-superflat with some $\bar{z}$.

Lemma 4.43. *Let p be a $\bar{z}$-flat program. Then there is an effectively computable $\bar{v}$-superflat program q such that*

$$\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \ \bar{v} \leftarrow q; \mathrm{ret}\, \bar{z} \tag{4.57}$$

where $\bar{v}$ is a vector containing all the variables (free and bound) occurring in q.

Proof. Let us first prove the claim under the assumption that p is cut-free. By Remark 4.21 we ensure that Kleene star occurs in p only in the form $(\mathrm{init}\, \bar{x} \leftarrow \mathrm{ret}\, \bar{x}\, \mathrm{in}\, r^\star)$. We take a $\bar{v}$ a vector of distinct variables containing all the variables occurring in p (free and bound). We construct q by induction over term complexity of p with respecto to the clauses of Definition 4.13.

FL1. If $p \doteq \varnothing$ then $q := \varnothing$. If $p \doteq \mathrm{ret}\, \bar{z}$ then $q := \mathrm{ret}\, \bar{v}$. If $p \doteq a_{0 \triangleright \bar{z}}$ then $q := a_{0 \triangleright \bar{v}}$. If $p \doteq a_{i \triangleright \bar{z}}$ and $i > 0$ then $q := a_{j \triangleright \bar{v}}$ where j is the number of z_i in $\bar{v}$. The induction invariant is trivially verified.

FL2. Let $p \doteq (u + r)$. By the induction hypothesis,

$$\vdash_{\mathsf{ME}^\star} u = \mathrm{do}\ \ \bar{v} \leftarrow u'; \mathrm{ret}\, \bar{z},$$
$$\vdash_{\mathsf{ME}^\star} r = \mathrm{do}\ \ \bar{v} \leftarrow r'; \mathrm{ret}\, \bar{z}$$

for some appropriate u', r'. Then we put $q := u' + r'$. The induction invariant is trivially verified.

FL3. Let $p \doteq (\mathrm{do}\ \bar{y} \leftarrow u; r)$. By the induction hypothesis,

$$\vdash_{\mathsf{ME}^\star} u = \mathrm{do}\ \bar{v} \leftarrow u'; \mathrm{ret}\,\bar{y},$$
$$\vdash_{\mathsf{ME}^\star} r = \mathrm{do}\ \bar{v} \leftarrow r'; \mathrm{ret}\,\bar{z}$$

for some appropriate u', r'. Let $q := (\mathrm{do}\ \bar{v} \leftarrow u'; r')$. Let us prove (4.57). We have

$$\begin{aligned}
p &\doteq \mathrm{do}\ \bar{y} \leftarrow u; r\\
&= \mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u'; \mathrm{ret}\,\bar{y}); \bar{v} \leftarrow r'; \mathrm{ret}\,\bar{z}\\
&= \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u'; \mathrm{ret}\,\bar{y}); r'); \mathrm{ret}\,\bar{z} && \text{[by Lem. 4.1]}\\
&= \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u'; \mathrm{ret}\,\bar{y}); \bar{v} \leftarrow \mathrm{ret}\,\bar{v}; r'); \mathrm{ret}\,\bar{z} && \text{[by } \textbf{(unit}_2\textbf{)}\text{]}\\
&= \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{y} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u'; \mathrm{ret}\,\bar{y});\\
&\quad\ \mathrm{ret}\,\bar{v}); r'); \mathrm{ret}\,\bar{z} && \text{[by Lem. 4.1]}\\
&= \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow u'; r'); \mathrm{ret}\,\bar{z} && \text{[by Lem. 4.26]}
\end{aligned}$$

FL4. Let $p \doteq (\mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}\,r^\star)$. By induction hypothesis, there is $\bar{v}$-flat program w all flat subterms of which have the same footprint and such that $\vdash_{\mathsf{ME}^\star} r = (\mathrm{do}\ \bar{v} \leftarrow w; \mathrm{ret}\,\bar{z})$. Let $q := (\mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,w^\star)$. Let us show (4.57). By induction hypothesis $\vdash_{\mathsf{ME}^\star} p = \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow w; \mathrm{ret}\,\bar{z})^\star$. Observe that

$$\vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{z} \leftarrow \mathrm{ret}\,\bar{z}; (\mathrm{do}\ \bar{v} \leftarrow w; \mathrm{ret}\,\bar{z}) = \mathrm{do}\ \bar{v} \leftarrow w; \mathrm{ret}\,\bar{z}$$

and $\mathrm{Vars}(\bar{v}) \cap \mathrm{Vars}(\mathrm{ret}\,\bar{z}) \subseteq \mathrm{Vars}(\bar{z})$. Therefore, by Corollary 4.8

$$\vdash_{\mathsf{ME}^\star} \mathrm{init}\,\bar{z} \leftarrow \mathrm{ret}\,\bar{z}\,\mathrm{in}(\mathrm{do}\ \bar{v} \leftarrow w; \mathrm{ret}\,\bar{z})^\star = \mathrm{init}\,\bar{v} \leftarrow \mathrm{ret}\,\bar{v}\,\mathrm{in}\,w^\star\ \mathrm{res}\,\bar{v} \to \mathrm{ret}\,\bar{z}$$

i.e. $\vdash_{\mathsf{ME}^\star} p = (\mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z})$ which completes the proof of the clause.

We have proved the claim under the assumption that p is cut-free. In general, if p is arbitrary $\bar{z}$-flat, by Lemma 4.28 we can find a cut-free $\bar{x}$-flat program w such that $\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{x} \leftarrow w; \mathrm{ret}\,\bar{z}$ and $\mathrm{Vars}(\bar{z}) \subseteq \mathrm{Vars}(\bar{x})$. Let q be a $\bar{v}$-superflat program such that $\vdash_{\mathsf{ME}^\star} w = \mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{x}$ whose existence we have ensured above. Note that $\mathrm{Vars}(\bar{x}) \subseteq \mathrm{Vars}(\bar{v})$. By Lemma 4.1, we have $\vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,\bar{z}$, and thus the proof is completed. $\square$

Let us recall the equivalence relation $\approx_{\mathcal{E}}$ defined on page 33. We shall now introduce a new equivalence relation $\simeq^{\sharp\mathcal{E}}_{\omega}$ over shallow-deterministic programs as the transitive closure of the union $(\approx_{\mathcal{E}} \cup \simeq^{\mathcal{E}}_{\omega})$. Note that by definition, $\simeq^{\sharp\mathcal{E}}_{\omega}$ is weaker than $\equiv_{\mathcal{E}}$ (page 60). We extend $\simeq^{\sharp\mathcal{E}}_{\omega}$ pointwise over $\mathcal{S}_\Sigma$ under the same name. Also, we denote by $\lesssim^{\sharp\mathcal{E}}_{\omega}$ the partial order induced by $\simeq^{\sharp\mathcal{E}}_{\omega}$, i.e. the one defined by the equivalence $P \lesssim^{\sharp\mathcal{E}}_{\omega} Q \iff P \cup Q \simeq^{\sharp\mathcal{E}}_{\omega} Q$.

Lemma 4.44. *Let p and q be two programs with the same computational return type. Then $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$ iff $\mathsf{NF}(p) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{NF}(q)$.*

Proof. Let $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$. Then $\mathsf{NF}(p) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{NF}(q)$ matches the conclusion of Lemma 3.36 under the assignments: $\Phi := \mathcal{E}$, $\bar{u} := \star$, $r := \mathrm{ret}\, y$ and $\simeq := \simeq_\omega^{\sharp\mathcal{E}}$. We are left to verify the premise, i.e. we need to show that for every $(s = t) \in \mathcal{E}$, all programs $u_1, \ldots, u_n, r$ and every normal $a \in \mathcal{A}_\Sigma$:

$$\mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[s/v]; r) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{NF}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}; y \leftarrow a[t/v]; r).$$

By (3.21), it suffices to prove the equivalence

$$\gamma(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[s/v]; r_i) \simeq_\omega^{\sharp\mathcal{E}} \gamma(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[t/v]; r_i) \tag{4.58}$$

for every i where $\bar{u}^i := \mathsf{unf}_i(\bar{u})$ and $r_i := \mathsf{unf}_i(r)$. Observe that for every i, by definition, $(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[s/v]; r_i) \approx_\mathcal{E} (\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[t/v]; r_i)$ and by Theorem 2.21:

$$\mathsf{nf}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[s/v]; r_i) \equiv_\mathcal{E} \mathsf{nf}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[t/v]; r_i).$$

Both sides of the latter equivalence are nondeterministic sums, which by definition of $\equiv_\mathcal{E}$ are elementwise equal under $\equiv_\mathcal{E}$. Application of prg to each of them returns the set of all their summands except those which contain Kleene star not under the ret. Therefore, $\mathsf{prg}(\mathsf{nf}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[s/v]; r_i))$ and $\mathsf{prg}(\mathsf{nf}(\mathrm{do}\ \bar{x} \leftarrow \bar{u}^i; y \leftarrow a[t/v]; r_i))$ must be elementwise equal under $\equiv_\mathcal{E}$. Since both the latter sets belong to $\mathcal{S}_\Sigma$, elementwise equivalence under $\equiv_\mathcal{E}$ can be replaced by elementwise equivalence under $\approx_\mathcal{E}$, and thus we arrive at (4.58) (recall that, by definition, γ is precisely $\mathsf{prg} \circ \mathsf{nf}$).

Now, suppose $\mathsf{NF}(p) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{NF}(q)$ and prove that $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$. By the completeness theorem, we will be done once we show that every strong continuous Kleene monad validating $\mathcal{E}$ also validates the equality $p = q$. Let $\mathbb{T}$ be a strong continuous Kleene monad validating $\mathcal{E}$. Then the sets $\mathsf{NF}(p)$ and $\mathsf{NF}(q)$ must be elementwise equal over it. By Lemma 3.38, $p = \sup \mathsf{NF}(p) = \sup \mathsf{NF}(q) = q$ over $\mathbb{T}$, and thus we are done. □

Given two vectors of variables $\bar{v}$ and $\bar{z}$, we call $\bar{z}$ a *clone* of $\bar{v}$ if $|\bar{z}| = |\bar{v}|$ and for every i, z_i and v_i are of the same type.

Lemma 4.45. *Let p and q be two deterministic $\bar{v}$-superflat programs, and let u and r be two arbitrary programs of the same return type. Let $\bar{v}'$ be a clone of $\bar{v}$ such that $\mathrm{Vars}(\bar{v}) \cap \mathrm{Vars}(\bar{v}') = \emptyset$. Then for any data theory $\mathcal{E}$ the equality*

$$\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\, u = \mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\, r \tag{4.59}$$

is provable in $\mathsf{ME}^\star$ *modulo* $\mathcal{E}$ *iff there is a substitution* $\sigma : \mathrm{Vars}(\bar{v}) \to \mathrm{Vars}(\bar{v}) \cup \mathrm{Vars}(\bar{v}')$ *such that* $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = r\sigma$ *and*

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} q = \mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{v}\sigma. \tag{4.60}$$

The same equivalence holds if $\mathsf{ME}^\star$ *replaced with* ME^ω.

Proof. Suppose for some substitution $\sigma : \mathrm{Vars}(\bar{v}) \to \mathrm{Vars}(\bar{v}) \cup \mathrm{Vars}(\bar{v}')$ we have (4.60) and $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = r\sigma$. Then (4.59) is provable in $\mathsf{ME}^\star$ modulo $\mathcal{E}$ as follows:

$$\begin{aligned}
&\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,u \\
&\quad = \mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,r\sigma \\
&\quad = \mathrm{do}\ \bar{v} \leftarrow p; \bar{v} \leftarrow \bar{v}\sigma; \mathrm{ret}\,r && [\text{by } \textbf{(unit}_2\textbf{)}] \\
&\quad = \mathrm{do}\ \bar{v} \leftarrow (\mathrm{do}\ \bar{v} \leftarrow p; \bar{v}\sigma); \mathrm{ret}\,r && [\text{by Lemma 4.1}] \\
&\quad = \mathrm{do}\ \bar{v} \leftarrow q; \mathrm{ret}\,r. && [\text{by 4.60}]
\end{aligned}$$

In order to prove the converse we ensure first that p is of the form:

$$\mathrm{do}\ \bar{v} \leftarrow a^1_{k_1 \triangleright \bar{v}}; \ldots; \bar{v} \leftarrow a^n_{k_n \triangleright \bar{v}}; \mathrm{ret}\,\bar{v}$$

by normalising $(\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{v})$, which is equivalent to p, by the associativity law and by further cancelling from the result all the program sequences of the form $\bar{v} \leftarrow \mathrm{ret}\,\bar{v}$. We also bring p to the analogous form. Assume that we have

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,u = \mathrm{do}\ \bar{v} \leftarrow q\varsigma; \mathrm{ret}\,r \tag{4.61}$$

where ς is some substitution $\mathrm{Vars}(\bar{v}) \to \mathrm{Vars}(\bar{v}) \cup \mathrm{Vars}(\bar{v}')$ and prove the existence of an analogous substitution σ such that $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = r\sigma$ and

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\,\bar{v}\sigma = q\varsigma. \tag{4.62}$$

This would complete the proof under the assignment $\varsigma := [\,]$. We proceed by induction over n. If $n = 0$ then (4.61) turns into $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{ret}\,u = \mathrm{ret}\,r\varsigma$, which implies $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = \mathrm{do}\ z \leftarrow \mathrm{ret}\,u; z = \mathrm{do}\ z \leftarrow \mathrm{ret}\,r\sigma; z = r\sigma$ and (4.62) under $\sigma := \varsigma$.

Consider the case $n > 0$. Let $p \doteq (\mathrm{do}\ \bar{v} \leftarrow a_{i \triangleright \bar{v}}; s)$. By Lemma 4.44, q must be of the form $(\mathrm{do}\ \bar{v} \leftarrow b_{j \triangleright \bar{v}}; t)$, where b is such that $\mathcal{E} \vdash_{\mathsf{EQ}} a = b\varsigma$. Let $z := v_j\varsigma$ and observe that $\mathcal{E} \vdash_{\mathsf{ME}^\star} q\varsigma = \mathrm{do}\ z \leftarrow b\varsigma; t\varsigma$. Therefore, (4.61) results in

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ v_i \leftarrow a; \bar{v} \leftarrow s; \mathrm{ret}\,u = \mathrm{do}\ z \leftarrow a; \bar{v} \leftarrow t\varsigma; \mathrm{ret}\,r. \tag{4.63}$$

First we consider the case $v_i \notin \mathrm{Vars}(t\varsigma)$. This assumption allows us to soundly rename the bound variable z in (4.63) to v_i by α-conversion. After that we obtain by Lemma 4.44: $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\,u = \mathrm{do}\ \bar{v} \leftarrow t\varsigma[v_i/z]; \mathrm{ret}\,r$. By induction hypothesis, there is an appropriate substitution σ such that $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, \bar{v}\sigma = t\varsigma[v_i/z]$ and $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = r\sigma$. We can prove (4.62) with the same σ as follows:

$$\begin{aligned}
&\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\, \bar{v}\sigma\\
&\quad = \mathrm{do}\ v_i \leftarrow a; \bar{v} \leftarrow s; \mathrm{ret}\, \bar{v}\sigma && \text{[by Lem. 4.1, } \textbf{(unit}_2\textbf{)}]\\
&\quad = \mathrm{do}\ v_i \leftarrow a; t\varsigma[v_i/z]\\
&\quad = \mathrm{do}\ z \leftarrow a; t\varsigma\\
&\quad = q\varsigma.
\end{aligned}$$

Now let us consider the case $v_i \in \mathrm{Vars}(t\varsigma)$. Since ς is a map from $\mathrm{Vars}(\bar{v})$ to $\mathrm{Vars}(\bar{v}) \cup \mathrm{Vars}(\bar{v}')$, by the pigeonhole principle, there is at least one variable from $\mathrm{Vars}(\bar{v}')$, say v, of the same type as v_i and such that $v \notin \mathrm{Vars}(t\varsigma)$. Hence we can switch by α-conversion from (4.63) to:

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ v \leftarrow a; \bar{v} \leftarrow s[v/v_i]; \mathrm{ret}\, u = \mathrm{do}\ v \leftarrow a; \bar{v} \leftarrow t\varsigma[v/z]; \mathrm{ret}\, r.$$

By Lemma 4.44, we have $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s[v/v_i]; \mathrm{ret}\, u = \mathrm{do}\ \bar{v} \leftarrow t\varsigma[v/z]; \mathrm{ret}\, r$. After applying to the both sides of the latter equation a substitution θ, swapping v and v_i we obtain $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, u = \mathrm{do}\ \bar{v} \leftarrow t\varsigma[v/z]\theta; \mathrm{ret}\, r$. By induction hypothesis there is an appropriate substitution σ such that $\mathcal{E} \vdash_{\mathsf{ME}^\star} u = r\varsigma[v/z]\theta$ and $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, \bar{v}\sigma = t\varsigma[v/z]\theta$. Let us apply θ to both sides of the latter equation. We thus obtain $\mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow s[v/v_i]; \mathrm{ret}\, \bar{v}\sigma = t\varsigma[v/z]$. Now we can prove (4.62) with the same σ as follows:

$$\begin{aligned}
&\mathrm{do}\ \bar{v} \leftarrow p; \mathrm{ret}\, \bar{v}\sigma\\
&\quad = \mathrm{do}\ v_i \leftarrow a; \bar{v} \leftarrow s; \mathrm{ret}\, \bar{v}\sigma && \text{[by Lem. 4.1, } \textbf{(unit}_2\textbf{)}]\\
&\quad = \mathrm{do}\ v \leftarrow a; \bar{v} \leftarrow s[v/v_i]; \mathrm{ret}\, \bar{v}\sigma\\
&\quad = \mathrm{do}\ v \leftarrow a; t\varsigma[v/z]\\
&\quad = \mathrm{do}\ z \leftarrow a; t\varsigma\\
&\quad = q\varsigma.
\end{aligned}$$

The proof of the analogous statement about provability in ME^ω runs analogously. □

Definition 4.46 (A-consistency)**.** We call the data theory $\mathcal{E}$ *A-consistent* for $A \in \mathrm{Type}_W^T$ if for any two distinct variables x, y of type A, the equality $x = y$ is not provable in $\mathcal{E}$.

Lemma 4.47. *Let p be a $\bar{v}$-superflat program, and let $\bar{A}$ be the return type of the vector $\bar{v}$. Suppose for some k, $\mathcal{E}$ is not A_k-consistent. Then for any vector of distinct variables $\bar{z}$ such that* $\mathrm{Vars}(\bar{z}) = \mathrm{Vars}(\bar{v}) \backslash \{v_k\}$, *there exists an effectively computable $\bar{z}$-flat program q such that*

$$\mathcal{E} \vdash_{\mathsf{ME}^\star} p = \mathrm{do}\ \bar{z} \leftarrow q; \mathrm{ret}\, \bar{v}. \tag{4.64}$$

Proof. Let us define $q := h(p)$ where h recursively transforms $\bar{v}$-superflat programs to $\bar{z}$-superflat ones according to the assignments respecting the clauses of Definition 4.13:

FL1. $h(\varnothing) := \varnothing; h(\operatorname{ret}\bar{v}) := \bar{z}; h(a_{i\triangleright\bar{v}}) := a_{j\triangleright\bar{z}}$ where j is the index of v_k in $\bar{z}$ if $k \neq 0, i \neq k$ and $j := 0$ otherwise.

FL2. $h(u + r) := h(u) + h(r)$.

FL3. $h(\operatorname{do}\ \bar{v} \leftarrow u; r) := \operatorname{do}\ \bar{z} \leftarrow h(u); h(r)$.

FL4. $h(\operatorname{init}\bar{v} \leftarrow u \operatorname{in} r^\star) := \operatorname{init}\bar{z} \leftarrow h(u) \operatorname{in} h(r)^\star$.

The provable equality (4.64) easily follows by induction over the complexity of p. We show it in the example case $p \doteq a_{k\triangleright\bar{v}}$, which effectively calls for A_k-inconsistency. We have $\mathcal{E} \vdash_{\mathsf{ME}^\star} p = a_{k\triangleright\bar{v}} = \operatorname{do}\ v_k \leftarrow a; \operatorname{ret}\bar{v}$. Since $\mathcal{E}$ is not A_k-consistent, the latter program is provably equal to $(\operatorname{do}\ a; \operatorname{ret}\bar{v})$. On the other hand, $\mathcal{E} \vdash_{\mathsf{ME}^\star} \operatorname{do}\ \bar{z} \leftarrow q; \operatorname{ret}\bar{v} = \operatorname{do}\ \bar{z} \leftarrow a_{0\triangleright\bar{z}}; \operatorname{ret}\bar{v} = \operatorname{do}\ a; \operatorname{ret}\bar{v}$ and thus we obtain (4.64). □

Lemma 4.48. *Let p and q be two superflat programs with the same footprint. Then the problems $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$ and $\mathcal{E} \vdash_{\mathsf{ME}^\star} p = q$ are equivalent and decidable.*

Proof. Let $\bar{z}$ be the common footprint of p, q, and let $\bar{A}$ be the return type of $\bar{z}$. By Lemma (4.47), we ensure A_k-consistency of $\mathcal{E}$ for every k.

We ensure that $\operatorname{Vars}(p) \subseteq \operatorname{Vars}(\bar{z})$ and $\operatorname{Vars}(q) \subseteq \operatorname{Vars}(\bar{z})$ by replacing all the variables from $\operatorname{Vars}(\bar{z})\backslash(\operatorname{Vars}(p) \cup \operatorname{Vars}(q))$. All the variables from the latter set obviously may not occur under ret, hence flatness is maintained under this transformation.

Let $\bar{z}'$ be a clone of $\bar{z}$ consisting of fresh variables. Let Q be the set of all atomic programs $a\sigma$ where a ranges over all atomic programs such that for some k, $a_{k\triangleright\bar{z}}$ occurs in one of the p_i, q_j and σ ranges over all variable substitutions from $\operatorname{Vars}(\bar{z}) \cup \operatorname{Vars}(\bar{z}')$ to itself. Let $Q_{\mathcal{E}}$ be the factor-set of Q modulo the provable equivalence in EQ with $\mathcal{E}$ as the set of axioms. For every $a \in Q$ let us denote by $[a]_{\mathcal{E}}$ the corresponding equivalence class from $Q_{\mathcal{E}}$. For every equivalence class $\mathcal{C}$ from $Q_{\mathcal{E}}$ we introduce a fresh signature symbol $f_{\mathcal{C}}$ with the typing $\bar{A} \times \bar{A} \to A$ where A is the common return type of programs from $\mathcal{C}$. For every $a \in Q$ let $a^\# := f_{[a]_{\mathcal{E}}}(\bar{z}, \bar{z}')$. More generally, we define the operator $\_^\#$ over $\bar{z}$-superflat programs as follows: for every t, $t^\#$ is obtained from t by replacing every $a_{k\triangleright\bar{z}}$ with $a^\#_{k\triangleright\bar{z}}$. For every $\bar{z}$-superflat program t let $\chi(t)$ be the $\bar{z}$-superflat program obtained from t by replacing every $f_{\mathcal{C}}(\bar{s}, \bar{r})_{k\triangleright\bar{z}}$ with $\sum_{s\in\mathcal{C}}(s[\bar{s}/\bar{z}, \bar{r}/\bar{z}'])_{k\triangleright\bar{z}}$. Obviously, for every $\bar{z}$-superflat t, in particular if $t \in \{p, q\}$, $\mathcal{E} \vdash_{\mathsf{ME}^+} \chi(t^\#) = t$. A somewhat less evident property can be proved by induction over term complexity of t:

$$\operatorname{tr}(\chi(t^\#)) = \bigcup\nolimits_{s\in\operatorname{tr}(t^\#)} \operatorname{tr}(\chi(s)). \tag{4.65}$$

Our goal now is to prove the implication:

$$\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q \implies \vdash_{\mathsf{ME}^\omega} \chi(p^\#) = \chi(q^\#). \tag{4.66}$$

Since by Lemma 4.31, $\vdash_{\mathsf{ME}^\omega} p = q$ is decidable and equivalent to $\vdash_{\mathsf{ME}^\star} p = q$ this would imply the claim by the circular implication:

$$\begin{aligned}\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q \implies \vdash_{\mathsf{ME}^\omega} \chi(p^\#) = \chi(q^\#) \implies \vdash_{\mathsf{ME}^\star} \chi(p^\#) = \chi(q^\#)\\ \implies \mathcal{E} \vdash_{\mathsf{ME}^\star} p = q \implies \mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q.\end{aligned}$$

Assume that $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$. Hence $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = \chi(q^\#)$ and by Lemmas 4.44, 4.11, we have:

$$\{\mathsf{nf}(s) \mid s \in \mathsf{tr}(p)\} \simeq_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(t) \mid t \in \mathsf{tr}(\chi(q^\#))\}. \tag{4.67}$$

Let us fix some $s \in \mathsf{tr}(p)$. By (4.67) and (4.65) there should exist $t \in \mathsf{tr}(q^\#)$ such that $\{\mathsf{nf}(s)\} \lesssim_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t))\}$. Let s' be the program obtained from s as follows: we normalise $(\mathsf{do}\ \bar{z} \leftarrow s; \mathsf{ret}\,\bar{z})$ under the associativity law, cancel all the program sections of the form $\bar{z} \leftarrow \mathsf{ret}\,\bar{z}$ and replace every program section of the form $\bar{z} \leftarrow a_{k \triangleright \bar{z}}$ by $z_k \leftarrow a$. Evidently $\mathsf{nf}(s) = \mathsf{nf}(s')$ and s' has form $(\mathsf{do}\ \bar{x} \leftarrow \bar{a}; \mathsf{ret}\,\bar{z})$ where $\mathrm{Vars}(\bar{x}) \subseteq \mathrm{Vars}(\bar{z})$ and all the a_i are atomic. In the same fashion we obtain t' from t.

Let V be a set of variables containing infinitely many variables of each type from Type_W^T and such that $\mathrm{Vars}(\bar{z}) \subseteq V$, $\mathrm{Vars}(\bar{z}') \cap V = \emptyset$. Let us show by induction over $n := |\bar{x}|$ that for any two injection $\sigma_1, \sigma_2 : \mathrm{Vars}(\bar{z}) \to V$:

$$\begin{aligned}&\{\mathsf{nf}(s'\sigma_1)\} \lesssim_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t'\sigma_2))\} \implies\\ &\{\mathsf{nf}(s'\sigma_1)\} \lesssim_\omega \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t'\sigma_2))\}.\end{aligned} \tag{4.68}$$

Assume that $n = 0$ and $\{\mathsf{nf}(s'\sigma_1)\} \lesssim_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t'\sigma_2))\}$. By Lemma 4.44, t' must be equal to $\mathsf{ret}\,\bar{z}$, and thus we have $\mathsf{ret}\,\bar{z}\sigma_1 \lesssim_\omega^{\sharp\mathcal{E}} \mathsf{ret}\,\bar{z}\sigma_2$. Observe that for every k, $z_k\sigma_1$ must be equal to $z_k\sigma_2$ — otherwise we establish an obvious contradiction with the assumption of A_k-consistency of $\mathcal{E}$. Now the right-hand side of (4.68) becomes obvious.

Assume that $n > 0$ and $\{\mathsf{nf}(s'\sigma_1)\} \lesssim_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t'\sigma_2))\}$. Let $s' \doteq (\mathsf{do}\ x \leftarrow a; u)$. By Lemma 4.44, t' must be of the form $(\mathsf{do}\ y \leftarrow b^\#; w)$ where $\mathcal{E} \vdash_{\mathsf{EQ}} a\sigma_1 = b\sigma_2$ and for some variable $v \in V \backslash (\mathrm{Vars}(u) \cup \mathrm{Vars}(r))$,

$$\{\mathsf{nf}(u[v/x]\sigma_1)\} \lesssim_\omega^{\sharp\mathcal{E}} \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(w[v/y]\sigma_2))\}.$$

Note that both $[v/x]\sigma_1$ and $[v/y]\sigma_2$ are again injective substitutions from $\mathrm{Vars}(\bar{z})$ to V. Therefore, by induction hypothesis, $\{\mathsf{nf}(u[v/x]\sigma_1)\} \lesssim_\omega \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(w[v/y]\sigma_2))\}$, and we will be done with the proof of (4.68) as soon as we show that $\vdash_{\mathsf{ME}^+} a\sigma_1 \leqslant \chi(b^\#\sigma_2)$. We proceed as follows: from $\mathcal{E} \vdash_{\mathsf{EQ}} a\sigma_1 = b\sigma_2$, we conclude that $\mathcal{E} \vdash_{\mathsf{EQ}} a\sigma_1\sigma_2^{-1} = b$. Consider the substitution $\sigma_1\sigma_2^{-1}$. It sends part of the variables from $\mathrm{Vars}(\bar{z})$ to $\mathrm{Vars}(\bar{z})$ and part of them to V. Let ς be a substitution sending every $v \in V$ such that for some k, $v = z_k(\sigma_1\sigma_2^{-1})$ to z'_k. We have $\mathcal{E} \vdash_{\mathsf{EQ}} a(\sigma_1\sigma_2^{-1}\varsigma) = b$ and the substitution $(\sigma_1\sigma_2^{-1}\varsigma)$ acts from $\mathrm{Vars}(\bar{z})$ to $\mathrm{Vars}(\bar{z}) \cup \mathrm{Vars}(\bar{z}')$. Both $a(\sigma_1\sigma_2^{-1}\varsigma)$ and b are elements of Q, hence they are members of the same equivalence class from $Q_{\mathcal{E}}$. Therefore $\vdash_{\mathsf{ME}^+} a\sigma_1\sigma_2^{-1}\varsigma \leqslant \chi(b^\#)$ and thus $\vdash_{\mathsf{ME}^+} a\sigma_1 \leqslant \chi(b^\#)\sigma_2\varsigma^{-1}$. Note that $\chi(b^\#)\sigma_2\varsigma^{-1} \doteq \chi(b^\#)\varsigma^{-1}\sigma_2$ and therefore we can rewrite the latter judgment as $\vdash_{\mathsf{ME}^+} a\sigma_1 \leqslant (\chi(b^\#)\varsigma^{-1})\sigma_2$. Since for every $c \in [b]_{\mathcal{E}}$, $\mathcal{E} \vdash_{\mathsf{EQ}} c = b$ and b does not contain variables from $\mathrm{Vars}(\bar{z}')$ for every such c, $\mathcal{E} \vdash_{\mathsf{EQ}} c\varsigma^{-1} = b$ and therefore $\vdash_{\mathsf{ME}^+} \chi(b^\#)\varsigma^{-1} = \chi(b^\#)$. Hence $\vdash_{\mathsf{ME}^+} a\sigma_1 \leqslant \chi(b^\#\sigma_2)$, and the proof of (4.68) is completed.

If we now instantiate σ_1 and σ_2 in (4.68) by the identity substitution, the premise becomes true, and thus we obtain $\{\mathsf{nf}(s')\} \lesssim_\omega \{\mathsf{nf}(r) \mid r \in \mathsf{tr}(\chi(t'))\}$. By Lemma 4.11 and Corollary 3.39, we have $\vdash_{\mathsf{ME}^\star} p \leqslant \chi(q^\#)$. Therefore, $\vdash_{\mathsf{ME}^\star} \chi(p^\#) \leqslant \chi(q^\#)$. By the symmetric argument, $\vdash_{\mathsf{ME}^\star} \chi(q^\#) \leqslant \chi(p^\#)$, and hence we are done. □

Theorem 4.49. *Let p and q be two weakly iterative programs with the same return type. Then the problems $\mathcal{E} \vdash_{\mathsf{ME}^\star} p = q$ and $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$ are equivalent. Moreover, if the conditional word problem for $\mathcal{E}$ is decidable, then both the problems are also decidable.*

Proof. We prove the claim by induction over the complexity of A. If A is T-free, then by Lemma 3.22, $\mathsf{nf}(p)$ and $\mathsf{nf}(q)$ do not contain any control operators, and thus both $\mathsf{NF}(p)$ and $\mathsf{NF}(q)$ are singletons containing $\mathsf{nf}(p)$ and $\mathsf{nf}(q)$ correspondingly. By Lemma 4.44, $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q$ implies $\mathsf{nf}(p) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{nf}(q)$. Therefore we are done by the following circular implication:

$$\mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q \implies \mathsf{nf}(p) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{nf}(q) \implies \mathcal{E} \vdash_{\mathsf{ME}^\star} p = q \implies \mathcal{E} \vdash_{\mathsf{ME}^\omega} p = q.$$

If $A = B \times C$ for some $B, C \in \mathrm{Type}_W^T$. Observe that

$$\begin{aligned}
\vdash_{\mathsf{ME}^\star} p = q &\iff \vdash_{\mathsf{ME}^\star} \mathrm{fst}(p) = \mathrm{fst}(q) \land \vdash_{\mathsf{ME}^\star} \mathrm{snd}(p) = \mathrm{snd}(q),\\
\vdash_{\mathsf{ME}^\omega} p = q &\iff \vdash_{\mathsf{ME}^\omega} \mathrm{fst}(p) = \mathrm{fst}(q) \land \vdash_{\mathsf{ME}^\omega} \mathrm{snd}(p) = \mathrm{snd}(q).
\end{aligned}$$

Since the return types of $\mathrm{fst}(p), \mathrm{fst}(q), \mathrm{snd}(p), \mathrm{snd}(q)$ are simpler than A, we are easily done by the induction hypothesis.

The remainder of the proof is devoted to the case $A = TB$. We shall only establish equivalence and decidability of $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$ and $\mathcal{E} \vdash_{\mathsf{ME}^\star} p \leqslant q$. This would evidently imply the goal by symmetry. By virtue of Lemmas 4.41 and 4.43, we assume w.l.o.g. that $p \doteq (\sum_i \mathrm{do}\ \bar{v} \leftarrow p_i; \mathrm{ret}\, r_i)$ and $q \doteq (\sum_j \mathrm{do}\ \bar{v} \leftarrow q_j; \mathrm{ret}\, u_j)$ where all the p_i, q_j are $\bar{v}$-superflat, all the r_i, u_j are weakly iterative and $\bar{v}$ encompasses all the variables occurring in the p_i, q_j.

Let $\bar{v}'$ be a clone of $\bar{v}$ consisting of fresh variables. For every i, j, let $\Xi_{i,j}$ be the set of all substitutions from $\mathrm{Vars}(\bar{v})$ to $\mathrm{Vars}(\bar{v}) \cup \mathrm{Vars}(\bar{v}')$ such that $\mathcal{E} \vdash_{\mathsf{ME}^\omega} r_i\sigma = u_j$. Note that by induction hypothesis $\mathcal{E} \vdash_{\mathsf{ME}^\omega} r_i\sigma = u_j$ is equivalent to $\mathcal{E} \vdash_{\mathsf{ME}^\star} r_i\sigma = u_j$ (since the return types of r_i and u_j are simpler than A) and is decidable.

Our plan is now as follows: first, we prove that $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$ implies

$$\forall i.\, \mathcal{E} \vdash_{\mathsf{ME}^\omega} p_i \leqslant \textstyle\sum_j \sum_{\sigma\in\Xi_{i,j}} \mathrm{do}\ \bar{v} \leftarrow q_j; \mathrm{ret}\, \bar{v}\sigma. \tag{4.69}$$

Second, we prove that (4.69) is equivalent to

$$\forall i.\, \mathcal{E} \vdash_{\mathsf{ME}^\star} p_i \leqslant \textstyle\sum_j \sum_{\sigma\in\Xi_{i,j}} \mathrm{do}\ \bar{v} \leftarrow q_j; \mathrm{ret}\, \bar{v}\sigma \tag{4.70}$$

and decidable. Finally, we show that (4.70) implies $\mathcal{E} \vdash_{\mathsf{ME}^\star} p \leqslant q$. Together with the evident implication $\mathcal{E} \vdash_{\mathsf{ME}^\star} p \leqslant q \implies \mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$ all this would result in equivalence and decidability of $\mathcal{E} \vdash_{\mathsf{ME}^\star} p \leqslant q$, $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$ and (4.70).

Suppose $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$. By Lemma 4.44, $\mathsf{NF}(p) \lesssim_\omega^{\sharp\mathcal{E}} \mathsf{NF}(q)$, which can be unfolded to

$$\textstyle\bigcup_i \{\mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, r_i) \mid s \in \mathsf{tr}(p_i)\} \lesssim_\omega^{\sharp\mathcal{E}} \bigcup_j \{\mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, u_j) \mid t \in \mathsf{tr}(q_j)\}. \tag{4.71}$$

Let us fix some i and some $s \in \mathsf{tr}(p_i)$. By (4.71), there exist j and $t \in \mathsf{tr}(q_j)$ such that $\mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, r_i) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, u_j)$. In particular,

$$\mathcal{E} \vdash_{\mathsf{ME}^\omega} \mathrm{do}\ \bar{v} \leftarrow s; \mathrm{ret}\, r_i = \mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, u_j.$$

Observe that the conditions of Lemma 4.45 are satisfied. Therefore for some $\sigma \in \Xi_{i,j}$, $\mathcal{E} \vdash_{\mathsf{ME}^\omega} s = \mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, \bar{v}\sigma$. By Theorem 2.21, $\mathsf{nf}(s) \equiv_{\mathcal{E}} \mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, \bar{v}\sigma)$ and therefore $\mathsf{nf}(s) \simeq_\omega^{\sharp\mathcal{E}} \mathsf{nf}(\mathrm{do}\ \bar{v} \leftarrow t; \mathrm{ret}\, \bar{v}\sigma)$. We have thus proved that for every i:

$$\mathsf{NF}(p_i) \lesssim_\omega^{\sharp\mathcal{E}} \textstyle\bigcup_j \bigcup_{\sigma\in\Xi_{i,j}} \mathsf{NF}(\mathrm{do}\ \bar{v} \leftarrow q_j; \mathrm{ret}\, \bar{v}\sigma)$$

from which (4.69) follows by Lemma 4.44.

Let us show that (4.69) implies (4.70). Let $h : \bar{A} \to \bar{A}$ be some fresh functional symbol where $\bar{A}$ is the return type of $\bar{v}$ correspondingly. Then (4.69) evidently implies

$$\forall i.\, \mathcal{E} \vdash_{\mathsf{ME}^\omega} \mathrm{do}\ \bar{v} \leftarrow p_i; h(\bar{v})_{0\triangleright\bar{v}} \leqslant \textstyle\sum_j \sum_{\sigma\in\Xi_{i,j}} \mathrm{do}\ \bar{v} \leftarrow q_j; h(\bar{v}\sigma)_{0\triangleright\bar{v}}. \tag{4.72}$$

If we treat the variables from $\bar{v}'$ occurring in the latter inequality as constants, we obtain $\bar{v}$-flat programs on both sides (in fact, even $\bar{v}$-superflat). Hence, by Lemma 4.48, (4.72) is equivalent to

$$\forall i.\, \mathcal{E} \vdash_{\mathsf{ME}^\star} \mathrm{do}\ \bar{v} \leftarrow p_i; h(\bar{v})_{0\triangleright\bar{v}} \leqslant \textstyle\sum_j \sum_{\sigma\in\Xi_{i,j}} \mathrm{do}\ \bar{v} \leftarrow q_j; h(\bar{v}\sigma)_{0\triangleright\bar{v}}.$$

and decidable. We can now obtain (4.70) by instantiating h with the ret operator.

Finally, let us show that (4.70) implies $\mathcal{E} \vdash_{\mathsf{ME}^\omega} p \leqslant q$. The proof of this fact is as follows:

$$\begin{aligned}
p &= \textstyle\sum_i \mathsf{do}\ \bar{v} \leftarrow p_i; \mathsf{ret}\, r_i \\
&\leqslant \textstyle\sum_i \mathsf{do}\ \bar{v} \leftarrow \Big(\sum_j \sum_{\sigma\in\Xi_{i,j}} \mathsf{do}\ \bar{v} \leftarrow q_j; \mathsf{ret}\, \bar{v}\sigma\Big); \mathsf{ret}\, r_i && \text{[by 4.70]} \\
&= \textstyle\sum_i \sum_j \sum_{\sigma\in\Xi_{i,j}} \mathsf{do}\ \bar{v} \leftarrow q_j; \bar{v} \leftarrow \mathsf{ret}\, \bar{v}\sigma; \mathsf{ret}\, r_i && \text{[by Lem. 4.1]} \\
&= \textstyle\sum_i \sum_j \sum_{\sigma\in\Xi_{i,j}} \mathsf{do}\ \bar{v} \leftarrow q_j; \mathsf{ret}\, r_i\sigma && \text{[by } \textbf{(unit}_2\textbf{)}\text{]} \\
&= \textstyle\sum_i \sum_j \sum_{\sigma\in\Xi_{i,j}} \mathsf{do}\ \bar{v} \leftarrow q_j; \mathsf{ret}\, u_j \\
&\leqslant q
\end{aligned}$$

The proof of the theorem is now completed. □

4.6 Contribution and related work

This chapter was entirely dedicated to the proof of the main positive result of the thesis (Theorem 4.49). We started with a completeness and decidability proof for a very limited fragment of MCE, called tight programs. We established a correspondence between tight programs and regular expression, and hence proved a completeness and decidability theorem by reduction to Kozen's completeness result [Koz94]. Then we successively extended the decidable and complete class of programs until we reached weakly-iterative programs. Comparing the definition of tight programs with the definition of weakly iterative programs provides a means to directly estimate the advance of expressivity when switching from the language of Kleene algebras to the language of strong Kleene monads, i.e. MCE.

A simple analysis of the presented proofs shows that the whole challenge is precisely due to the combination of two factors: the return operator and multivariable contexts (i.e. the syntactic counterpart of monadic strength). It can easily be seen that in the absence of the return operator, Lemma 4.23 would close the issue altogether. On the other hand, in the absence of strength, MCE can support only programs with precisely one variable (cf. e.g. [Mog91]), and thus most of our proofs would simply collapse.

As we have argued, our completeness and decidability result can be viewed as a counterpart of Kozen's result regarding the completeness of Kleene algebra over the algebra of regular events. It is known that Kleene algebra is also complete with respect to so-called relational interpretation, i.e. the interpretation over an algebra of binary relations [KS96]. In terms of monads, the analogue of relational interpretation is the powerset monad (its Kleisli category is precisely the category of sets and relation). Notably, strong Kleene monads are incomplete with respect to the powerset monad, since in contrast to Kleene algebras one can express in MCE (actually, already in ME) specific properties of powerset monad falsified by other strong Kleene monads such as symmetry [Koc70]:

$$\mathrm{do}\ x \leftarrow p; y \leftarrow q; \mathrm{ret}\langle x, y\rangle = \mathrm{do}\ y \leftarrow q; x \leftarrow p; \mathrm{ret}\langle x, y\rangle.$$

One can expect that the calculus of a strong (continuous) Kleene monad is still complete over some reasonably narrow class of monads. An expected candidate to this effect is the class of state powerset monads.

Chapter 5

Conclusions

In this thesis we have addressed the problem of logical treatment of computational effects with nondeterminism and iteration. We have chiefly followed the approach originally developed by Moggi in his seminal works [Mog89b, Mog91]. Every particular notion of computation we consider thus as part of a package: a monad class, a (variant of) computational metalanguage, a sound and complete calculus. Our results regarding computational monads are commonly presented and proved in terms of an appropriate metalanguage that greatly facilitates both the presentation and the calculations. Our method can be characterised as a combination of rewriting techniques and (fixed-point) induction.

In Chapter 1 we justified the settings for the research that followed. We formulated a classical result on soundness and completeness of the metalanguage of effects over strong monads as a prototype for the similar theorems that followed. Our contribution includes a novel strongly normalising rewriting system complete over strong monads. We introduced a notion of plain signature and data theory as a compromise variant of background axiomatisation and extended the decidability theorem for the provable equality modulo data theories.

In Chapter 2 we introduced the class of strong additive monads as a lightweight device for computational effects featuring nondeterminism as a component. Our main motivation for this particular notion of nondeterminism was to capture the same laws that are standard for propositional dynamic logic, thus enabling the use of Fischer-Ladner encoding in order to derive a branching operator [FL79]. We outlined how new strong additive monads can be obtained from existing ones. We introduced a confluent and strongly normalising system of reduction completely capturing the axioms of strong additive monads except the ACI-laws for nondeterministic choice. Based on this result, we proved a Church-Rosser theorem stating that two nondeterministic programs are equal if their normal forms are ACI-equal. These results were suitably extended for the case of a background data theory.

In Chapter 3 we introduced two important classes of monads: strong Kleene monads and strong continuous Kleene monads, sharing the same language which we called the metalanguage of control and effects (MCE). Compared to existing notions of monads featuring both fixed points and nondeterminism (cf. e.g. [EG03, Jac10]), our notion of strong Kleene monads has the strongest parallels with the concept of Kleene algebra. In pursuing this similarity we introduced the notion of strong continuous Kleene monads as a fair analogue of the $*$-continuous Kleene algebra (cf. e.g. [Koz94]). Intuitively, strong continuous Kleene monads can be considered as standard strong Kleene monads, just like $*$-continuous Kleene algebras are considered as standard Kleene algebras. We started the investigation of the relationship between strong Kleene monads and strong continuous Kleene monads by introducing an infinitary calculus for strong continuous Kleene monads and proving a number of metalogical properties of it, which led us to a non-r.e. result. As a consequence, the calculus of strong Kleene monads turned out to be (equationally) incomplete over strong continuous Kleene monads. Using similar ideas, we proved that the calculus of strong Kleene monads is undecidable.

In Chapter 4 we concentrated on the fragments of MCE with decidable program equality. We introduced a hierarchy of program classes (see Appendix B) and proved a completeness and decidability theorem for each of them by successive reduction from one to the other. Our main result is that the equality of two weakly iterative programs modulo a data theory is decidable and is completely captured by the proof calculus for strong Kleene monads. Informally, weakly iterative programs can be described as those where a Kleene star does not encode self-substitutions of the return operators.

Chapter 6

Further work

An evident next step to be made on top of the achieved decidability results is the introduction of *assertions*, special programs with $T1$ as the return type and capable of blocking the execution of a program in respect of a logical condition. An analogue of this already exists in the realm of Kleene algebra under the name *Kleene algebra with tests* [KS96, Koz97]. This would allow us to encode the conventional imperative programming operators as follows:

$$\begin{aligned} \mathit{if}(b,p,q) &:= \mathrm{do}\ b?; p + \mathrm{do}\ (\neg b)?; q, \\ \mathit{while}(b,p) &:= \mathrm{init}\, x \leftarrow \mathrm{ret}\, x\, \mathrm{in}(\mathrm{do}\ b?; p)^{\star}\, \mathrm{res}\, x \to (\mathrm{do}\ (\neg b)?; \mathrm{ret}\, x) \end{aligned}$$

where '?' is a special operator, returning $\mathrm{ret}\,\star$ for true statements and $\varnothing$ for false ones.

Integration of our results regarding the equational logic for MCE with the research on monad-based dynamic logics [SM03, SM04, GSM06, MSG10] appears to be a promising perspective. We expect that many issues concerned with decidability may disappear due to reduction of the expressive power. On the other hand, we might need to develop some sort of hypothesis elimination procedure (cf. [Coh94, KS96, HK02a]) in order to be able to prove logical entailment.

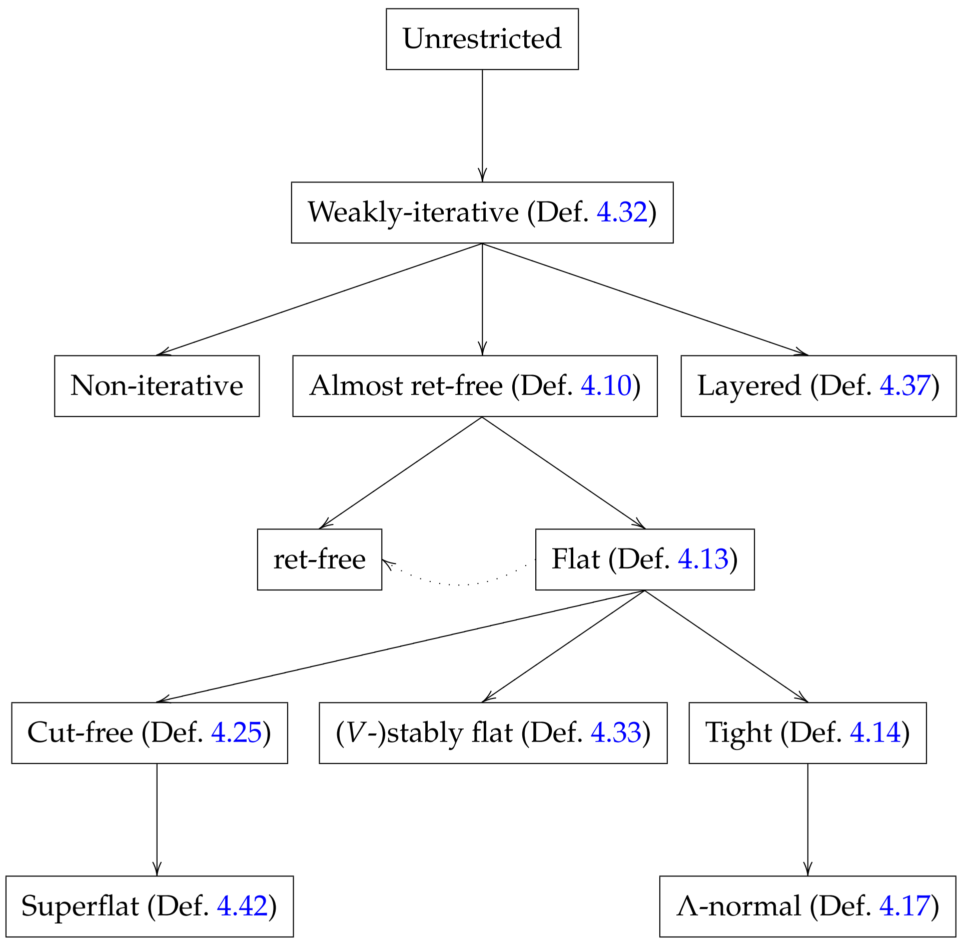
The class of strong (continuous) pseudo-Kleene monads rather superficially outlined in Remark 3.43 seems to be quite interesting because, on one hand, all the important results valid for strong (continuous) Kleene monads seem to remain valid for it, and on the other hand, it is broad enough to include some interesting monads which fail to be Kleene. Another motivation comes from the conceptual side: the definition of Kleene monads ensures two independent effects — nontermination and nondeterminism — and hence, as we have seen, is not an unbreakable unit. On the other hand, one can take a strong (continuous) pseudo-Kleene monad as a basis and introduce an advanced exception mechanism on top of it, e.g. like the one described in [SM04]. We leave the detailed investigation of pseudo-Kleene monads as a subject for further research.

Whereas Kleene monads can be used as a model for recursive computations with effects, doing effectful recursion does not necessarily imply nondeterminism. In fact, introducing nondeterminism in order to obtain a (pseudo-)Kleene monad might present considerable difficulties. Unfortunately, there is no generic robust method to empower a monad with nondeterminism, but we hope that one can find a way to embed a monad implementing a sufficiently general notion of recursion into a Kleene monad. This would enable us to assume w.l.o.g. that recursive programs are always interpreted over Kleene monads; in particular, this might help us to reuse our completeness results for purely deterministic settings.

Appendix A: Summary of reductions

Name	Rule set
$\star$	$(p : E) \ \rightarrowtail\ e_E \qquad (p \neq e_E)$
β	$\mathrm{fst}\langle p, q\rangle \ \rightarrowtail\ p \qquad \mathrm{snd}\langle p, q\rangle \ \rightarrowtail\ q$ $\mathrm{do}\ x \leftarrow \mathrm{ret}\, p; q \ \rightarrowtail\ q[p/x]$ $\mathrm{do}\ x \leftarrow (\mathrm{do}\ y \leftarrow p; q); r \ \rightarrowtail\ \mathrm{do}\ y \leftarrow p; x \leftarrow q; r \quad (y \notin FV(r))$
η	$\langle \mathrm{fst}(p), e_E\rangle \ \rightarrowtail\ p \qquad \langle e_E, \mathrm{snd}(p)\rangle \ \rightarrowtail\ p$ $\mathrm{do}\ x \leftarrow p; \mathrm{ret}\, e_E \ \rightarrowtail\ p \quad \mathrm{do}\ x \leftarrow p; \mathrm{ret}\, x \ \rightarrowtail\ p$ $\langle \mathrm{fst}(p), \mathrm{snd}(p)\rangle \ \rightarrowtail\ p$
σ	$p + \varnothing \ \rightarrowtail\ p \qquad \mathrm{do}\ x \leftarrow p; \varnothing \ \rightarrowtail\ \varnothing$ $\varnothing + p \ \rightarrowtail\ p \qquad \mathrm{do}\ x \leftarrow \varnothing; p \ \rightarrowtail\ \varnothing$ $\mathrm{do}\ x \leftarrow (p + q); r \ \rightarrowtail\ \mathrm{do}\ x \leftarrow p; r + \mathrm{do}\ x \leftarrow q; r$ $\mathrm{do}\ x \leftarrow p; (q + r) \ \rightarrowtail\ \mathrm{do}\ x \leftarrow p; q + \mathrm{do}\ x \leftarrow p; r$
k	$\mathrm{init}\, x \leftarrow p\, \mathrm{in}\, q^\star \ \rightarrowtail\ p + \mathrm{init}\, x \leftarrow (\mathrm{do}\ x \leftarrow p; q)\, \mathrm{in}\, q^\star$

Appendix B: Relations between several important program classes



The arrows are directed from the larger class to the smaller one.
The only dotted arrow means ‘larger modulo the first unit law’.

Bibliography

[AHK07] Kamal Aboul-Hosn and Dexter Kozen. Local variable scoping and Kleene algebra with tests. *J. Log. Algebr. Program.*, 2007.

[AKKB99] Egidio Astesiano, Hans-Joerg Kreowski, and Bernd Krieg-Brueckner, editors. *Algebraic Foundations of Systems Specification*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.

[AMV10] Jiří Adámek, Stefan Milius, and Jiří Velebil. Equational properties of iterative monads. *Math. Structures in Comput. Sci.*, 2010. (accepted).

[BBdP98] Nick Benton, Gavin M. Bierman, and Valeria de Paiva. Computational types from a logical perspective. *J. Funct. Prog.*, 8(2):177–193, 1998.

[BD86] Leo Bachmair and Nachum Dershowitz. Commutation, transformation, and termination. In *Proc. of the 8th international conference on Automated deduction*, pages 5–20, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[BE93] Stephen L. Bloom and Zoltán Ésik. *Iteration theories: the equational logic of iterative processes*. Springer-Verlag New York, Inc., New York, NY, USA, 1993.

[BKdV03] Marc Bezem, Jan Willem Klop, and Roel de Vrijer, editors. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[BW85] Michael Barr and Charles Wells. *Toposes, Triples and Theories*, volume 278 of *Grundlehren der mathematischen Wissenschaften*. Springer, New York, 1985.

[Cd96] Pierre-Louis Curien and Roberto di Cosmo. A confluent reduction for the lambda-calculus with surjective pairing and terminal object. *J. Funct. Program.*, 6(2):299–327, 1996.

[CJ10] Dion Coumans and Bart Jacobs. Scalars, monads, and categories. (unpublished), 2010.

[CM93] Pietro Cenciarelli and Eugenio Moggi. A syntactic approach to modularity in denotational semantics. Technical report, In Proceedings of the Conference on Category Theory and Computer Science, 1993.

[Coh94] Ernie Cohen. (unpublished), 1994.

[CP90] Roy L. Crole and Andrew M. Pitts. New foundations for fixpoint computations. In John C. Mitchell, editor, *Logic in Computer Science, LICS 1990*, pages 489–497. IEEE, 1990.

[Cro94] Roy L. Crole. *Categories for Types (Cambridge Mathematical Textbooks)*. Cambridge University Press, January 1994.

[DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Comm. ACM*, 22:465–476, 1979.

[EG03] Patrik Eklund and Werner Gähler. Partially ordered monads and powerset kleene algebras. In *7th Seminar RelMiCS, 2nd Workshop Kleene Algebra*, pages 40–42, 2003.

[EL00] Levent Erkök and John Launchbury. Recursive monadic bindings. In *ICFP'00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, pages 174–185, New York, NY, USA, 2000. ACM.

[Fil94] Andrzej Filinski. Representing monads. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Programming Languages*, pages 446–457. ACM Press, 1994.

[FL79] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 1979.

[FM06] Matthew Fluet and Greg Morrisett. Monadic regions. *J. Funct. Program.*, 16(4-5):485–545, 2006.

[FMA06] Matthew Fluet, Greg Morrisett, and Amal J. Ahmed. Linear regions are all you need. In *Programming Languages and Systems, ESOP 2006*, volume 3924 of *LNCS*, pages 7–21. Springer, 2006.

[GM81] Joseph A. Goguen and José Meseguer. Completeness of many-sorted equational logic. *SIGPLAN Not.*, 16(7):24–32, 1981.

[Gro94] Philippe de Groote. Strong normalization in a non-deterministic typed lambda-calculus. In *LFCS '94: Proceedings of the Third International Symposium on Logical Foundations of Computer Science*, pages 142–152, London, UK, 1994. Springer-Verlag.

[GSM06] Sergey Goncharov, Lutz Schröder, and Till Mossakowski. Completeness of global evaluation logic. In *MFCS*, pages 447–458, 2006.

[GTA95] Wolfgang Gehrke, Phd Thesis, and Osterreichischer Akademischer Austauschdienst. Decidability results for categorical notions related to monads by rewriting techniques, 1995.

[GTL89] Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*. Cambridge University Press, New York, NY, USA, 1989.

[Hin64] J. R. Hindley. *The Church-Rosser Property and a Result in Combinatory Logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.

[Hin00] Ralf Hinze. Deriving backtracking monad transformers. *ACM SIGPLAN Notices*, 35(9):186–197, 2000.

[HJS06] Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace theory. In *International Workshop on Coalgebraic Methods in Computer Science (CMCS 2006), volume 164 of Elect. Notes in Theor. Comp. Sci*, pages 47–65. Elsevier, 2006.

[HK02a] Chris Hardin and Dexter Kozen. On the elimination of hypotheses in Kleene algebra with tests. Technical Report TR2002-1879, Computer Science Department, Cornell University, October 2002.

[HK02b] Masahito Hasegawa and Yoshihiko Kakutani. Axioms for recursion in call-by-value. *Higher Order Symbol. Comput.*, 15(2-3):235–264, 2002.

[HLPP07] Martin Hyland, Paul Blain Levy, Gordon Plotkin, and John Power. Combining algebraic effects with continuations. *Theoretical Computer Science*, 375(1-3):20 – 40, 2007. Festschrift for John C. Reynolds's 70th birthday.

[HPP03] Martin Hyland, Gordon Plotkin, and John Power. Combining effects: Sum and tensor. *Theoretical Computer Science*, 2003.

[Jac10] Bart Jacobs. From coalgebraic to monoidal traces. In *Coalgebraic Methods in Computer Science*. Elsevier, 2010. (to appear).

[JM84] Jean-Pierre Jouannaud and Miguel Munoz. Termination of a set of rules modulo a set of equations. In *Automated Deduction, CADE 1984*, volume 170 of *LNCS*, pages 175–193. Springer, 1984.

[JP03] Bart Jacobs and Erik Poll. Coalgebras and Monads in the Semantics of Java. *Theoret. Comput. Sci.*, 291:329–349, 2003.

[Klo92] Jan W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 1–117. Oxford University Press, 1992.

[Koc70] Anders Kock. Monads on symmetric monoidal closed categories. *Archiv der Mathematik*, 21:1–10, 1970.

[Koc72] Anders Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972.

[Koz94] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Inf. Comput.*, 110:366–390, 1994.

[Koz96] Dexter Kozen. Kleene algebra with tests and commutativity conditions. In *Tools and Algorithms for Construction and Analysis of Systems, TACAS 1996*, LNCS, pages 14–33. Springer, 1996.

[Koz97] Dexter Kozen. Kleene algebra with tests. *Transactions on Programming Languages and Systems*, 19(3):427–443, May 1997.

[KS96] Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In D. van Dalen and M. Bezem, editors, *Proc. 10th Int. Workshop Computer Science Logic (CSL'96)*, volume 1258 of *Lecture Notes in Computer Science*, pages 244–259, Utrecht, The Netherlands, September 1996. Springer-Verlag.

[KSFS05] Oleg Kiselyov, Chung-chieh Shan, Daniel Friedman, and Amr Sabry. Backtracking, interleaving, and terminating monad transformers. In *Functional Programming, ICFP 2005*, pages 192–203. ACM, 2005.

[LG02] Christoph Lüth and Neil Ghani. Composing monads using coproducts. *SIGPLAN Not.*, 37(9):133–144, 2002.

[LHJ95] Sheng Liang, Paul Hudak, and Mark Jones. Monad transformersand modular interpreters. In *In Proceedings of the 22nd ACM Symposium on Principles of Programming Languages. ACMPress*, 1995.

[LS86] Joachim Lambek and Philip J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge, 1986.

[LS05] Sam Lindley and Ian Stark. Reducibility and $\top\top$-lifting for computation types. In *Typed Lambda Calculi and Applications: Proceedings of the Seventh International Conference TLCA 2005*, number 3461 in Lecture Notes in Computer Science, pages 262–277. Springer-Verlag, 2005.

[Man76] Ernest G. Manes. *Algebraic Theories*, volume 26 of *Graduate Texts in Mathematics*. Springer-Verlag, 1976.

[Mat67] Juri V. Matijasevič. Simple examples of undecidable associative calculi. *Soviet Mathematics (Dokladi)*, 8(2):555–557, 1967.

[ML71] Saunders Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer-Verlag, 1971.

[MN98] Richard Mayr and Tobias Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(192), 1998.

[Mog89a] Eugenio Moggi. An abstract view of programming languages. Technical Report ECS-LFCS-90-113, Edinburgh Univ., Dept. of Comp. Sci., June 1989. Lecture Notes for course CS 359, Stanford Univ.

[Mog89b] Eugenio Moggi. Computational lambda-calculus and monads. In *Proceedings 4th Annual IEEE Symp. on Logic in Computer Science, LICS'89, Pacific Grove, CA, USA, 5–8 June 1989*, pages 14–23. IEEE Computer Society Press, Washington, DC, 1989.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93:55–92, 1991.

[Mog95] Eugenio Moggi. A semantics for evaluation logic. *Fund. Inform.*, 22:117–152, 1995.

[MSG10] Till Mossakowski, Lutz Schröder, and Sergey Goncharov. A generic complete dynamic logic for reasoning about purity and effects. In J. Fiadeiro, editor, *Fundamental Approaches to Softeware Engineering (FASE 2008)*, Lecture Notes in Computer Science. Springer, 2010. (to appear).

[Pap98] Nikolaos S. Papaspyrou. *A Formal Semantics for the C Programming Language*. PhD thesis, National Technical University of Athens, 1998.

[Pit91] Andrew Pitts. Evaluation logic. In *Higher Order Workshop*, Workshops in Computing, pages 162–189. Springer, 1991.

[PJW93] Simon L. Peyton-Jones and Philip Wadler. Imperative functional programming. In *POPL '93: Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 71–84, New York, NY, USA, 1993. ACM.

[PM00] Nikolaos Papaspyrou and Dragan Macos. A study of evaluation order semantics in expressions with side effects. *J. Funct. Program.*, 10(3):227–244, 2000.

[PP02] Gordon Plotkin and John Power. Notions of computation determine monads. In *Proc. FOSSACS 2002, Lecture Notes in Computer Science 2303*, pages 342–356. Springer, 2002.

[PW02] John Power and Hiroshi Watanabe. Combining a monad and a comonad. *Theor. Comput. Sci.*, 280(1-2):137–162, 2002.

[Rog87] Hartley Rogers, Jr. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, USA, 1987.

[Ros90] Kimmo I. Rosenthal. *Quantales and their applications*. Number 234 in Pitman Research Notes in Mathematics Series. Longman Scientific & Technical, Harlow, 1990.

[RS97] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages, vol. 1: Word, Language, Grammar*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[SM03] Lutz Schröder and Till Mossakowski. Monad-independent Hoare logic in HASCASL. In *Fundamental Aspects of Software Engineering*, volume 2621 of *LNCS*, pages 261–277, 2003.

[SM04] Lutz Schröder and Till Mossakowski. Generic exception handling and the Java monad. In Charles Rattray, Savitri Maharaj, and Carron Shankland, editors, *Algebraic Methodology and Software Technology*, volume 3116 of *Lecture Notes in Computer Science*, pages 443–459. Springer Berlin, 2004.

[SP00] Alex Simpson and Gordon Plotkin. Complete axioms for categorical fixed-point operators. In *In Proceedings of 15th Annual Symposium on Logic in Computer Science*, pages 30–41, 2000.

[Str72] Ross Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2:149–168, 1972.

[Str06] Georg Struth. Abstract abstract reduction. *J. Log. Algebr. Program.*, 66(2):239–270, 2006.

[Wad92] Philip Wadler. The essence of functional programming. In *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–14. ACM Press, 1992.

[Wad94] Philip Wadler. Monads and composable continuations. *Lisp and Symbolic Computation*, 7(1):39–56, 1994.

[Wad95] Philip Wadler. Monads for functional programming. In J. Jeuring and E. Meijer, editors, *Advanced Functional Programming*, volume 925 of *Lecture Notes in Computer Science*, pages 24–52. Springer, 1995.