Theory and Applications of Categories, Vol. 10, No. 12, 2002, pp. 248–300.

SOBER SPACES AND CONTINUATIONS

PAUL TAYLOR

ABSTRACT. A topological space is sober if it has exactly the points that are dictated by its open sets. We explain the analogy with the way in which computational values are determined by the observations that can be made of them. A new definition of sobriety is formulated in terms of lambda calculus and elementary category theory, with no reference to lattice structure, but, for topological spaces, this coincides with the standard lattice-theoretic definition. The primitive symbolic and categorical structures are extended to make their types sober. For the natural numbers, the additional structure provides definition by description and general recursion.

We use the same basic categorical construction that Thielecke, Führmann and Selinger use to study continuations, but our emphasis is completely different: we concentrate on the fragment of their calculus that *excludes* computational effects, but show how it nevertheless defines new *denotational* values. Nor is this "denotational semantics of continuations using sober spaces", though that could easily be derived.

On the contrary, this paper provides the underlying λ-calculus on the basis of which abstract Stone duality will re-axiomatise general topology. The leading model of the new axioms is the category of locally compact locales and continuous maps.

Contents

1	Computational values	248
2	The restricted λ-calculus	255
3	Algebras and homomorphisms	259
4	Sobriety and monadicity	263
5	Topology revisited	267
6	Enforcing sobriety	272
7	The structure of $\mathsf{S}\mathcal{C}$	277
8	A lambda calculus for sobriety	281
9	Theory of descriptions	285
10	Sobriety and description	289
11	Directions	292

1. Computational values

What does it mean for a computation to yield a value?

If the computational object is a function, or a database measured in terabytes, we may only obtain *parts* of its value, by querying it with arguments or search-terms. It is usual to say that if the *type* of the object is simple then the object is directly observable, but

Received by the editors 2002 January 21 and, in revised form, 2002 July 3.

Transmitted by Andrew Pitts. Published on 2002 July 22.

2000 Mathematics Subject Classification: 06D22, 06E15, 18B30, 18C20, 18C50, 22A26, 54A05, 54C35, 54D10, 54D45.

Key words and phrases: observable type; predicate transformer; continuation passing style; sober space; locally compact space; locally quasi-compact space; locale; prime filter; Sierpinski space; schizophrenic object; Stone duality; strong monad; Kleisli category; premonoidal category; prime lambda term; sober lambda calculus; theory of descriptions; general recursive function.

for complex types we must perform some computational experiment in order to access the value.

Typically, $\mathbb{N}$ is regarded as an observable type [Plo77], but, as Alan Turing had already observed [Tur35, Section 8], if we are given two numbers with a lot of digits, say 9999999999999999 and 999999999999999, we may only determine whether or not they are equal by carefully comparing them digit by digit. For *very* large numbers, it may not even be feasible to print out *all* of the digits, so we are back in the situation of merely being prepared to print (or, indeed, to compute) whichever of the digits are actually required. Recursion theory traditionally regards the contents of a database as a huge number too.

So much for integers. What does it mean to define a *real* number? It is no good writing it out in decimal notation — even overlooking the ambiguity between 0.99999... and 1.00000... — because such an expression is necessarily finite, and therefore defines a *rational* number. For me to give you a real number in this notation, you have first to tell me *how many* decimal digits you require.

This *interactive* manner of obtaining mathematical values goes back to Weierstrass's definition of continuity of $f : \mathbb{R} \to \mathbb{R}$ at u,

$$\forall \epsilon > 0.\, \exists \delta > 0.\, \forall u'.\, |u' - u| < \delta \Rightarrow |f(u') - f(u)| < \epsilon.$$

We ask the *consumer* of $f(u)$ how much accuracy (ϵ) is required, and pass this information back to the *producer* of u as our own demand (δ) on the input.

1.1. Remark. In all of these examples, the value can *only* be elucidated by being ready to use it in situations that ultimately result in an observable value. In general, the best I can do is to *be prepared* (with a program) to provide *as much information as you actually require.*

The theme of this paper is that, once we have loosened our control over computational values to this extent, we open the floodgates to *many* more of them.

As $\mathbb{N}$ is too big a type to be observable, maybe we should use **2**, the type of bits? But no, this assumes that all computations terminate, so we need a type that's simpler still. The type Σ of ***semi-bits*** is the only observable type that we need: such a value may be present ("yes"), or may never appear ("wait"). Σ is like the type that is called `unit` in ML, but `void` in C and Java. A program of this type returns no useful information besides the fact that it has terminated, but it need not even do that. The results of many such programs may be used in parallel to light up a dot-matrix display, and thereby give an answer that is intelligible to humans.

1.2. Remark. Abstractly, it is therefore enough to consider a *single* program of type Σ, so long as we allow processing to go on in parallel.

A computation $\phi[x]$ of type Σ is an ***affirmative*** property of x, that is, a property that will (eventually) announce its truth, if it is true. Steven Vickers has given a nice account of properties that are affirmative but false, refutative but true, *etc.*, showing how the algebra of affirmative properties has finite conjunctions and infinitary disjunctions, just like the lattice of open subsets of a topological space [Vic88, Chapter 2].

Indeed, by running two processes in parallel and waiting for one or both of them to terminate, this algebra admits binary conjunction and disjunction, whilst there are trivial programs that denote $\bot$ and $\top$. The other possibility is to start off (one at a time) a lot of copies of the same program, giving them each the input 0, 1, 2, ..., and wait to see if one of them terminates. If the nth program terminates after N steps, we ignore (silently kill off) the $n+N-1$ other processes that we have started, and don't bother to start process number $n+N+1$ that's next in the queue to go. This ***existential quantifier*** is similar to the *search* or *minimalisation* operator in general recursion, though in fact it is simpler, and general recursion can be defined from it.

1.3. Definition. Mathematically, these constructions make Σ into a lattice with infinitary joins, over which meets $(\top, \wedge)$ distribute. It is convenient to consider finite $(\bot, \vee)$ and directed ($\bigvee^{\uparrow}$) joins separately. Allowing joins of arbitrary families, as is required in traditional point-set topology, such a lattice is called a ***frame***.

For computation, the joins must be recursively defined, and in particular countable. It is one of the objectives of the programme (Abstract Stone Duality) to which this paper is an introduction to re-formulate topology to agree with computation.

Because of the halting problem, there is no negation or implication. Nor is a predicate of the form $\forall n{:}\mathbb{N}.\ \phi[n]$ affirmative, as we never finish testing $\phi[n]$s. Whilst we can use proof theory to investigate stronger logics, we can only *talk about* them: the connectives $\wedge$ and $\vee$, and the quantifier $\exists n$, constitute the logic that we can *do*. In particular, we can *do* the pattern-matching and searching that proof theory needs by using $\wedge$, $\vee$ and $\exists$.

We write Σ^X for the type (lattice) of observations that can be made about values of type X, because λ-abstraction and application conveniently express the formal and actual roles of the value in the process of observation. Observations, being computational objects, are themselves values that we can access only by making observations of them. The type of meta-observations is called Σ^{Σ^X}, and of course there are towers of any height you please.

There is a *duality* between values and observations.

1.4. Remark. One special way of making a meta-observation $P : \Sigma^{\Sigma^X}$ about an observation $\phi : \Sigma^X$ is to apply it to a particular value $p : X$. We write

$$P = \eta_X(p) \quad \text{for the meta-observation with} \quad P(\phi) = \phi(p).$$

Thus P is a summary of the results $\phi(p)$ of *all* of the (possible) observations ϕ that we could make about p. (Being itself a computational object, the value of P can only be accessed by making observations ...)

If someone gives us a P, they are allegedly telling us the results of all of the observations that we might make about some value p, but to which they are giving us no direct access. Must we accept their word that there really is some value p behind P?

First, there are certain "healthiness" conditions that P must satisfy [Dij76, Chapter 3]. These are rather like testing the plausibility of someone's alibis: was it really possible for someone to have been in these places at these times?

1.5. Remark. The application of observations to a value p respects the lattice operations on the algebra of observations:

truth: If the observation $(\lambda x.\, \top)$ applied to the value p were *not* to terminate, this would mean that the computation of p did not terminate.

falsity: If $(\lambda x.\, \bot)p$ were to terminate, this would mean that the code to make the observation $(\lambda x.\, \bot)$ had never been executed: somehow the computation of p has *hijacked* the output channel of the observation. This is indeed done in various programming languages, with a command `abort`, `halt`, *etc.*, or by throwing an exception that is not caught. This raises the question of the *scope* of the exception: how far out of the execution environment is it *actually* caught, given that it doesn't bring the World to an end? Hayo Thielecke calls values that respect these constant observations ***discardable***, though the point is that it is *safe* to calculate them, even when we may not need to use them.

binary conjunction: If we can observe both $\phi(p)$ and $\psi(p)$ separately, then we can also observe $\phi(p)\wedge\psi(p)$. A computation p that changes the state or consumes some resource can fail this property by using *angelic* non-determinism: in the execution of $\phi(p)$, it so makes its internal choices as to make $\phi(p)$ more likely to succeed, but as it might want to make different choices for $\psi(p)$, it may not be able to win twice.

binary disjunction: If we can observe $\phi(p) \vee \psi(p)$, then we could instead observe either $\phi(p)$ or $\psi(p)$. A program p can fail this property by using *demonic* non-determinism. Suppose that $\phi(p)$ and $\psi(p)$, run individually, are unsuccessful, *i.e.* they don't terminate. However, if instead we run them in parallel, the way in which each of them changes the state amounts to *communication* between them, with the effect that the combined computation $\phi(p) \vee \psi(p)$ may follow a different and successful computation path.

directed joins: If we can observe $\bigvee^{\uparrow}_i \phi_i(p)$ then we can already observe some $\phi_i(p)$. This argument about finiteness of computation is familiar from the Rice–Shapiro theorem and 1970s domain theory.

See also [Smy94, Section 4.4] for further discussion of the relationship between non-determinism and the preservation of lattice structure.

1.6. Examples. Here are some programs that violate the properties above, *i.e.* which respectively fail to preserve the logical connectives on the left, although we are mis-using "the" here (Section 9).

$$
\begin{array}{lll}
\top & \mathsf{force}(\lambda\phi.\, \bot) & \mathsf{the}\ n.\ \bot \\
\bot & \mathsf{force}(\lambda\phi.\, \top) & \\
\wedge & \mathsf{force}(\lambda\phi.\, \phi[0] \vee \phi[1]) & \mathsf{the}\ n.\ (n = 0 \vee n = 1) \\
\vee & \mathsf{force}(\lambda\phi.\, \phi[0] \wedge \phi[1]) &
\end{array}
$$

1.7. Definition. A subset P of a frame is called a ***filter*** if it contains $\top$, it is closed upwards, and also under finite meets ($\wedge$). A ***completely coprime filter*** is one such that, if $\bigvee U \in P$, then already $u \in P$ for some $u \in U$. Classically, the complement of such a filter is a ***prime ideal***, I, which is closed downwards and under infinitary joins, $\top \notin I$, and if $u \wedge v \in I$ then either $u \in I$ or $v \in I$.

1.8. Remark. The motivations that we have given were translated from topology, using the dictionary

point	value
open subset	observation
open neighbourhood	observation of a value.

This view of general topology is more akin to Felix Hausdorff's approach [Hau14] in terms of the family of open neighbourhoods of each point than to the better known Bourbaki axiomatisation of the lattice of all open subsets of the space [Bou66]. Beware that we only consider *open* neighbourhoods, whereas for Bourbaki *any* subset is a neighbourhood so long as it contains an open subset around the point. Bourbaki writes $\mathfrak{B}(x)$ for the collection of such neighbourhoods of x.

1.9. Remark. The family $P = \eta_X(p)$ of open neighbourhoods of a point $p \in X$ is also a completely coprime filter in the frame Σ^X of open subsets of X:
(a) it is closed upwards, *i.e.* if $\phi \in P$ and $\phi \leq \psi$ then $\psi \in P$;
(b) $\top \in P$;
(c) it is closed under intersection: if $\phi, \psi \in P$ then $\phi \wedge \psi \in P$;
(d) it is coprime: $\bot \notin P$, and if $\phi \vee \psi \in P$ then either $\phi \in P$ or $\psi \in P$;
(e) it is Scott-open: if $\bigvee\!\!\!\!\nearrow_i \phi_i \in P$ then $\phi_i \in P$ for some i.

1.10. Remark. In Theorem 5.12 we make use of something that fails some of these conditions, namely the collection $\{U \mid K \subset U\}$ of open neighbourhoods of a *compact* subspace $K \subset X$. This is still a Scott-open filter (it respects $\top$, $\wedge$ and $\bigvee\!\!\!\!\nearrow$), but is only coprime if K is a singleton. (At least, that is the situation for T_1-spaces: the characterisation is more complicated in general. When we use this idea in Theorem 5.12, K must also be an upper subset in the specialisation order.)

1.11. Remark. So far we have only discussed computations that run on their own, without any input. In general, a program will take inputs $u_1 : U_1$, ..., $u_k : U_k$ over certain types, and we conventionally use Γ to name this list of typed variables. For the moment, we take $k = 1$.

Suppose that $P(u) : \Sigma^{\Sigma^X}$ is a meta-observation of type X that satisfies the conditions that we have described, for each input value $u \in U$. If $\phi : \Sigma^X$ is an observation of the output type X then $P(u)(\phi)$ is an observation of the input u, which we call $\psi(u) \equiv$

$H(\phi)(u)$. Then the lattice-theoretic properties of $P(u)$ transfer to H:

$$\begin{array}{rclcrcl} H(\top_{\Sigma^X}) & = & \top_{\Sigma^U} & \qquad & H(\phi_1 \wedge \phi_2) & = & H(\phi_1) \wedge H(\phi_2) \\ H(\bot_{\Sigma^X}) & = & \bot_{\Sigma^U} & & H(\phi_1 \vee \phi_2) & = & H(\phi_1) \vee H(\phi_2) \end{array}$$

together with the infinitary version, $H(\bigvee^{\uparrow} \phi_i) = \bigvee^{\uparrow} H(\phi_i)$.

1.12. Remarks.

(a) Such an $H : \Sigma^X \to \Sigma^U$ is called a ***frame homomorphism***, and any continuous function $f : U \to X$ defines such a homomorphism by $\Sigma^f : \phi \mapsto \lambda u.\ \phi(fu)$.

(b) This is the Bourbaki definition of continuity: for every open subset $\phi \subset X$ of the output, the inverse image, $\psi \equiv \Sigma^f \phi \equiv f^{-1}(\phi) \equiv \{u \mid fu \in \phi\} \subset U$, is an open subset of the input.

(c) In particular, for $f : \mathbb{R} \to \mathbb{R}$, $u \in \mathbb{R}$ and $\epsilon > 0$, let $\phi \equiv \{x \mid |x - fu| < \epsilon\}$ be an open interval around fu, then ψ is an open neighbourhood of u iff it contains the open interval $\{u' \mid |u' - u| < \delta\}$ for some $\delta > 0$. This is Weierstrass's definition.

(d) Computationally, if you tell me *what you're going to do with my output* (your ***continuation*** ϕ from my procedure), I can tell you *what* (ψ) *we're going to do with your input.*

So computations are given in the same contravariant way as continuous functions are defined in general topology.

1.13. Remark. Since we *only* access values *via* their observations,

if $\phi[a] = \phi[b]$ for all observations $\phi : \Sigma^X$ then $a = b : X$.

This is a ***Leibniz principle for values***. The corresponding property for points and open subsets of a topological space is known as the ***T_0 separation axiom***. An equality such as $\phi[a] = \phi[b]$ of two terms of type Σ means that one program terminates if and only if the other does. This equality is not itself an observable computation, as we cannot see the programs (both) failing to terminate.

1.14. Remark. Now suppose that the system P of observations does satisfy the consistency conditions that we have stated, *i.e.* it is a completely coprime filter. Must there now be some point $p \in X$ such that $P = \eta_X(p)$?

Sobriety says that there is — and then T_0 says that it's unique.

In the parametric version, every frame homomorphism $H : \Sigma^X \to \Sigma^U$ is given by Σ^f for some unique continuous function $f : U \to X$.

We shall show in this paper that the lattice-theoretic way in which we have introduced sobriety is equivalent to an equational one in the λ-calculus. In Section 9 we return to a lattice-theoretic view of $\mathbb{N}$, where the corresponding notion is that of a ***description***, *i.e.* a predicate that provably has exactly one witness. Then sobriety *produces* that witness, *i.e.* the number that is defined by the description. Taking the same idea a little further, we obtain the search operation in ***general recursion***.

In topology, sobriety says that spaces are determined (up to isomorphism) by their frames of open subsets, just as points are determined (up to equality) by their neighbourhoods. Sobriety is therefore a ***Leibniz principle for spaces***. The next step is to say that not only the spaces but the entire category of spaces and continuous functions is determined by the category of frames and homomorphisms — a ***Leibniz principle for categories***. This is developed in [B], for which we set up the preliminaries here.

Another idea, called ***repleteness***, was investigated in synthetic domain theory [Hyl91, Tay91]. This played the same role in the theory as sobriety (*cf.* Remark 10.9), but it is technically weaker in some concrete categories.

1.15. Remark. We have stressed that a meta-observation $P : \Sigma^{\Sigma^X}$ *only* defines a value of type X when certain conditions are satisfied. Indeed, we justified those conditions by excluding certain kinds of programs that have non-trivial ***computational effects***.

Since fire burns, we adopt precautions for avoiding it or putting it out — that is the point of view of this paper. On the other hand, fire is useful for cooking and heating, so we also learn how to use it safely.

The mathematical techniques discussed in this paper are closely related to those that have been used by Hayo Thielecke [Thi97a, Thi97b], Carsten Führmann [Füh99] and Peter Selinger [Sel01] to study computational effects. More practically, Guy Steele [Ste78] and Andrew Appel [App92] showed how an ordinary functional program $f : U \to X$ (without jumps, *etc.*) may be compiled very efficiently by regarding it as a continuation-transformer $\Sigma^X \to \Sigma^U$. This is called the ***continuation-passing style***. It may be extended to handle imperative idioms such as jumps, exceptions and co-routines by *breaking* the rules that we lay down. As in Remark 1.5, programs may *hijack* their continuations — altering them, not running them at all, or even calling them twice! We discuss this briefly in Remark 11.4.

Theoretical computer science often displays this ambiguity of purpose — are we applying mathematics to computation or *vice versa*? It is important to understand, of this and each other study, which it is trying to do.

The development of mathematics before Georg Cantor was almost entirely about the employment of computation in the service of mathematical ideas, but in an age of networks mathematics must now also be the servant of the science of complex systems, with non-determinism and computational effects. This paper and the programme that it introduces seek to use computational ideas as a foundation for conceptual mathematics. The science of systems is a travelling companion, but our destinations are different. This does not mean that our objectives conflict, because the new mathematics so obtained will be better suited than Cantor's to the denotational foundations of high-level computation.

2. The restricted λ-calculus

Although we have used completely coprime filters to introduce sobriety, we shall not use lattice theory in the core development in this paper, except to show in Section 5 that various categories of topological spaces and continuous functions provide models of the abstract structure.

We shall show instead that sobriety has a new characterisation in terms of the exponential $\Sigma^{(-)}$ and its associated λ-calculus. The abstract construction in Sections 3, 4, 6, 7 and 8 will be based on some category $\mathcal{C}$ about which we assume *only* that it has finite products, and powers $\Sigma^{(-)}$ of some special object Σ. In most of the applications, especially to topology, this category is *not* cartesian closed: it is only the object Σ that we require to be ***exponentiating***.

This structure on the category $\mathcal{C}$ may alternatively be described in the notation of the λ-calculus. When $\mathcal{C}$ is an already given concrete category (maybe of topological spaces, domains, sets or posets), this calculus has an *interpretation* or *denotational semantics* in $\mathcal{C}$. Equally, on the other hand, $\mathcal{C}$ may be an abstract category that is manufactured from the symbols of the calculus. The advantage of a categorical treatment is, as always, that it serves both the abstract and concrete purposes equally well.

2.1. Definition. The ***restricted λ-calculus*** has just the type-formation rules

$$\mathbf{1}\ \mathsf{type} \qquad \frac{X_1\ \mathsf{type} \quad \ldots X_k\ \mathsf{type}}{\Sigma^{X_1\times\cdots\times X_k}\ \mathsf{type}}\,\Sigma^{(-)}F$$

but with the normal rules for λ-abstraction and application,

$$\frac{\Gamma, x : X \vdash \sigma : \Sigma^Y}{\Gamma \vdash \lambda x{:}X.\ \sigma : \Sigma^{X\times Y}}\,\Sigma^{(-)}I \qquad \frac{\Gamma \vdash \phi : \Sigma^{X\times Y} \quad \Gamma \vdash a : X}{\Gamma \vdash \phi[a] : \Sigma^Y}\,\Sigma^{(-)}E$$

together with the usual α, β and η rules.

The turnstile ($\vdash$) signifies a sequent presentation in which there are all of the familiar structural rules: identity, weakening, exchange, contraction and cut.

2.2. Remark. As this is a fragment of the simply typed λ-calculus, it strongly normalises. We shall take a *denotational* view of the calculus, in which the β- and η-rules are *equations* between different notations for *the same value*, and are applicable at any depth within a λ-expression. (This is in contrast to the way in which λ-calculi are made to agree with the execution of programming languages, by restricting the applicability of the β-rule [Plo75]. Besides defining *call by name* and *call by value* reduction strategies, this paper used continuations to interpret one dually within the other.)

2.3. Notation. We take account of the restriction on type-formation by adopting a convention for variable names: lower case Greek letters and *capital* italics denote terms whose type is (a retract of) some Σ^X. These are the terms that can be the bodies of λ-abstractions. Since they are also the terms to which the lattice operations and $\exists$ below may be applied, we call them ***logical terms***. Lower case italic letters denote terms of arbitrary type.

As we have already seen, towers of Σs like Σ^{Σ^X} tend to arise in this subject. We shall often write $\Sigma^2 X$, and more generally $\Sigma^n X$, for these. (Fortunately, we do not often use finite discrete types, but when we do we write them in **bold**: $\mathbf{0}$, $\mathbf{1}$, $\mathbf{2}$, $\mathbf{3}$.) Increasingly exotic alphabets will be used for terms of these types, including

$$a, b, x, y : X \qquad \phi, \psi : \Sigma^X \qquad F, G : \Sigma^{\Sigma^X} \qquad \mathcal{F}, \mathcal{G} : \Sigma^3 X.$$

2.4. Remark. In order to model general topology (Section 5) we must add the lattice operations $\top$, $\bot$, $\wedge$ and $\vee$, with axioms to say that Σ is a distributive lattice. In fact, we need a bit more than this. The ***Euclidean principle***,

$$\phi : \Sigma,\ F : \Sigma^\Sigma \ \vdash\ \phi \wedge F(\phi) \ = \ \phi \wedge F(\top),$$

captures the extensional way in which Σ^X is a set of subsets [C]. It will be used for computational reasons in Proposition 10.6, and Remark 4.11 explains why this is necessary.

2.5. Remark. We shall also consider the type $\mathbb{N}$ of natural numbers, with primitive recursion at all types, in Sections 9–10 (where the lattice structure is also needed). Terms of this type are, of course, called ***numerical***. Note that $\mathbb{N}$ is a discrete set, not a domain with $\bot$. Strong normalisation is now lost.

The notation that we use for primitive recursion at type X is

$$\frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash z : X \qquad \Gamma,\ m : \mathbb{N},\ u : X \ \vdash\ s(m, u) : X}{\Gamma \ \vdash\ \mathsf{rec}\big(n, z, \lambda mu.\, s(m, u)\big) : X}$$

where Γ and $m : \mathbb{N}$ are static and dynamic parameters, and u denotes the "recursive call". The β-rules are

$$\mathsf{rec}(0, z, \lambda mu.\, s) = z \qquad \mathsf{rec}(n+1, z, \lambda mu.\, s) = s\big(n, \mathsf{rec}(n, z, \lambda mu.\, s)\big).$$

Uniqueness of the rec term is enforced by the rule

$$\frac{\Gamma \ \vdash\ z = r(0) \qquad \Gamma,\ m : \mathbb{N} \ \vdash\ s\big(m, r(m)\big) = r(m+1)}{\Gamma,\ n : \mathbb{N} \ \vdash\ \mathsf{rec}\big(n, z, \lambda mu.\, s(m, u)\big) = r(n),}$$

whose ingredients are exactly the base case and induction step in a traditional proof by induction.

Countable joins in Σ^Γ may be seen logically in terms of the existential quantifier

$$\frac{\Gamma, n : \mathbb{N} \ \vdash\ \phi[n] : \Sigma}{\Gamma \ \vdash\ \exists n.\, \phi[n] : \Sigma}$$

for which distributivity is known as the *Frobenius law*.

2.6. Remark. For both topology and recursion, a further axiom (called the ***Scott principle*** in [Tay91]) is also needed to force all maps to preserve directed joins:

$$\Gamma,\ F : \Sigma^2\mathbb{N}\ \vdash\ F(\lambda n.\, \top)\ =\ \exists n.\, F(\lambda m.\, m < n).$$

For any $\Gamma \vdash G : \Sigma^X \to \Sigma^X$, let $\Gamma\ \vdash\ YG\ =\ \exists n.\, \mathsf{rec}(n, \lambda x.\, \bot, \lambda m\phi.\, G\phi) : \Sigma^X$ and

$$\Gamma,\ x : X \vdash\ F\ =\ \lambda\phi.\, G\big(\exists n.\, \phi n \wedge \mathsf{rec}(n, \lambda x.\, \bot, \lambda m\phi.\, G\phi)\big)x.$$

Then $F(\lambda n.\, \top) = G(YG)x$ and $\exists n.\, F(\lambda m.\, m < n) = YGx$, so YG is a fixed point of G, indeed the least one. However, this axiom is *not* needed in this paper, or indeed until we get rather a long way into the abstract Stone duality programme [E, F].

2.7. Remark. We shall want to pass back and forth between the restricted λ-calculus and the corresponding category $\mathcal{C}$. The technique for doing this fluently is a major theme of [Tay99]; see Sections 4.3 and 4.7 in particular.

When a sequent presentation such as ours has all of the usual structural rules (in particular weakening and contraction), there is a category with products

- whose *objects* are the contexts (lists of typed variables), and
- whose *morphisms* are generated by
  - weakenings $\hat{x} : [\Gamma, x : X] \longrightarrow\!\!\triangleright\ \Gamma$ for each type X in context Γ, and
  - cuts $[a/x] : \Gamma \to [\Gamma, x : X]$ for each term $\Gamma \vdash a : X$;
- composites of these generating morphisms obey the (extended) substitution lemma.
- In the case of the (restricted) λ-calculus, the terms a that appear in the cut morphisms are λ-expressions *modulo* the α, β and η rules, and
- the language of these terms is extended for the extra structure in Remarks 2.4ff by the lattice connectives and laws, primitive recursion and the existential quantifier.

We shall not need the notation $\hat{x}$ in this paper, but we shall re-use the $\hat{}$ for a different purpose in Section 6. (There we also introduce a category $\mathsf{H}\mathcal{C}$ that does not have products, but, unlike other authors, we shy away from using a syntactic calculus to work in it, because such a calculus would have to specify an order of evaluation.)

2.8. Proposition. *The category $\mathcal{C}$ so described is the free category with finite products and an exponentiating object Σ, together with the additional lattice and recursive structure according to the context of the discussion.*

Proof. The mediating functor $[\![-]\!] : \mathcal{C} \to \mathcal{D}$ from this syntactic category $\mathcal{C}$ to another ("semantic") category $\mathcal{D}$ equipped with the relevant structure is defined by structural recursion. ∎

The exponentiating object Σ immediately induces certain structure in the category.

2.9. Lemma. *$\Sigma^{(-)}$ is a contravariant functor. In particular, for $f : X \to Y$, $\psi : \Sigma^Y$ and $F : \Sigma^2 X$, we have*

$$\Sigma^f(\psi) = \lambda x.\, \psi[fx] \qquad \textit{and} \qquad \Sigma^2 f(F) = \lambda\psi.\, F\big(\lambda x.\, \psi[fx]\big).$$

Proof. You can check that $\Sigma^{\mathsf{id}} = \mathsf{id}$ and $\Sigma^{f;g} = \Sigma^g ; \Sigma^f$. ∎

In general topology and locale theory it is customary to write $f^*\psi \subset X$ for the inverse image of $\psi \subset Y$ under f, but we use $\Sigma^f\psi$ instead for this, considered as a λ-term, saving f^* for the meta-operation of substitution (in Lemmas 8.7 and 9.2).

Now we can describe the all-important neighbourhood-family $\eta_X(x)$ (Remark 1.4) in purely categorical terms. As observed in [Tay99, Remark 7.2.4(c)], it is most unfortunate that the letter η has well established meanings for two different parts of the anatomy of an adjunction.

2.10. Lemma. *The family of maps $\eta_X : X \to \Sigma^{\Sigma^X}$, defined by $x \mapsto (\lambda\phi.\, \phi x)$, is natural and satisfies $\eta_{\Sigma^X} ; \Sigma^{\eta_X} = \mathsf{id}$ (which we call the **unit equation**).*

$$\begin{tikzcd}
\Sigma^2 X \arrow[r, "\Sigma^2 f"] & \Sigma^2 Y \\
X \arrow[u, "\eta_X"] \arrow[r, "f"] & Y \arrow[u, "\eta_Y"']
\end{tikzcd}
\qquad
\begin{tikzcd}
& \Sigma^3 X \arrow[dr, "\Sigma^{\eta_X}"] & \\
\Sigma^X \arrow[ur, "\eta_{\Sigma^X}"] \arrow[rr, "\mathsf{id}"] & & \Sigma^X
\end{tikzcd}$$

Proof. As this Lemma is used extremely frequently, we spell out its proof in the λ-calculus in detail. Using the formulae that we have just given,

$$\Sigma^2 f(\eta_X x) = \lambda\psi.\, (\lambda\phi.\, \phi x)\big(\lambda x'.\, \psi(fx')\big) = \lambda\psi.\, \big(\lambda x'.\, \psi(fx')\big)(x) = \lambda\psi.\, \psi(fx) = \eta_Y(fx).$$

Also, $\Sigma\eta_X(\mathcal{F}) = \lambda x.\, \mathcal{F}(\eta_X x) = \lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)$ for $\mathcal{F} : \Sigma^3(X)$, so

$$\Sigma\eta_X(\eta_{\Sigma X}\phi) = \lambda x.\, (\lambda F.\, F\phi)(\lambda\phi'.\, \phi' x) = \lambda x.\, (\lambda\phi'.\, \phi' x)(\phi) = \lambda x.\, \phi x = \phi.$$ ∎

2.11. Proposition. *The contravariant functor $\Sigma^{(-)}$ is symmetrically adjoint to itself on the right, the unit and counit both being η. The natural bijection*

$$\frac{H : X \longrightarrow \Sigma^\Gamma}{\overline{\overline{P : \Gamma \longrightarrow \Sigma^X}}}$$

*defined by $P = \eta_\Gamma ; \Sigma^H$ and $H = \eta_X ; \Sigma^P$ is called **double exponential transposition**.*

Proof. The triangular identities are both $\eta_{\Sigma^X} ; \Sigma^{\eta_X} = \mathsf{id}$. ∎

2.12. REMARK. The task for the next three sections is to characterise, in terms of category theory, lambda calculus and lattice theory, those $P : \Sigma^{\Sigma^X}$ that (should) arise as $\eta_X(a)$ for some $a : X$.

The actual condition will be stated in Corollary 4.12, but, whatever it is, suppose that the morphism or term

$$\Gamma \xrightarrow{P} \Sigma^{\Sigma^X} \qquad \text{or} \qquad \Gamma \vdash P : \Sigma^{\Sigma^X}$$

does satisfy it. Then the intuitions of the previous section suggest that we have defined a new value, which we shall call

$$\Gamma \vdash \mathsf{focus}\, P : X,$$

such that the result of the observation $\phi : \Sigma^X$ is

$$\Gamma \ \vdash \ \phi(\mathsf{focus}\, P) \ = \ P\phi : \Sigma.$$

In particular, when $P = \eta_X(a)$ we recover $a = \mathsf{focus}\, P$ and $\phi a = P\phi$. These are the β- and η-rules for a new constructor focus that we shall add to our λ-calculus in Section 8.

2.13. REMARK. In the traditional terminology of point-set topology, a completely coprime filter *converges* to its *limit* point, but the word "limit" is now so well established with a completely different meaning in category theory that we need a new word.

Peter Selinger [Sel01] has used the word "focus" for a category that is essentially our $\mathsf{S}\mathcal{C}$. The two uses of this word may be understood as *singular* and *collective* respectively: Selinger's focal subcategory consists of the legitimate results of our focus operator.

We now turn from the type X of values to the corresponding algebra Σ^X of observations, in order to characterise the homomorphisms $\Sigma^Y \to \Sigma^X$ that correspond to functions $X \to Y$, and also which terms $P : \Sigma^{\Sigma^X}$ correspond to virtual values in X.

3. Algebras and homomorphisms

3.1. NOTATION. The adjunction in Proposition 2.11 gives rise to a strong monad with
(a) multiplication $\mu_X = \Sigma^{\eta_{\Sigma^X}} : \Sigma^4 X \to \Sigma^2 X$ given by $\mathcal{F} \mapsto \lambda\phi.\, \mathcal{F}(\lambda F.\, F\phi)$, and
(b) strength $\sigma_{\Gamma,X} : \Gamma \times \Sigma^2 X \to \Sigma^2(\Gamma \times X)$ given by $\gamma, F \mapsto \lambda\psi.\, F\big(\lambda x.\, \psi(\gamma, x)\big)$,
satisfying six equations. Eugenio Moggi [Mog91] demonstrated how strong monads can be seen as notions of computation, giving a ("let") calculus in which μ is used to interpret composition and σ to substitute for parameters. Constructions similar to ours can be performed in this generality.

However, rather than develop an *abstract* theory of monads, our purpose is to demonstrate the relevance of one *particular* monad and show how *it* accounts for the intuitions in Section 1. The associativity law for μ involves $\Sigma^6 X$, but we certainly don't want to compute with such λ-terms unless it is absolutely necessary! In fact we can largely avoid using σ and μ in this work.

3.2. Definition. An ***Eilenberg–Moore algebra*** for the monad is an object A of $\mathcal{C}$ together with a morphism $\alpha : \Sigma^2 A \to A$ such that $\eta_A \,;\, \alpha = \mathsf{id}_A$ and $\mu_A \,;\, \alpha = \Sigma^2\alpha \,;\, \alpha$.

$$\begin{tikzcd}
A & \Sigma^2 A \arrow[l, "\alpha"'] \\
\Sigma^2 A \arrow[u, "\alpha"] & \Sigma^4 A \arrow[l, "\mu_A"] \arrow[u, "\Sigma^2\alpha"']
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \Sigma^2 A \arrow[dr, "\alpha"] & \\
A \arrow[ur, "\eta_A"] \arrow[rr, "\mathsf{id}"] & & A
\end{tikzcd}$$

3.3. Lemma. *For any object X, $(\Sigma^X, \Sigma^{\eta_X})$ is an algebra.*

Proof. The equations are just those in Lemma 2.10, although the one involving μ is $\Sigma^{(-)}$ applied to the naturality equation for η with respect to η_X. ∎

These are the *only* algebras that will be used in *this* paper, but [B] shows how general algebras may be regarded as the topologies on subspaces that are defined by an axiom of comprehension. In other words, all algebras are of this form, but with a generalised definition of the type X.

3.4. Definition. We shall show that the following are equivalent (when $A = \Sigma^X$ *etc.*):

(a) For any two algebras (A, α) and (B, β), a $\mathcal{C}$-morphism $H : B \to A$ is called an (***Eilenberg–Moore***) ***homomorphism*** if $\beta \,;\, H = \Sigma^2 H \,;\, \alpha$, as in the square on the left.

$$\begin{tikzcd}
A & \Sigma^2 A \arrow[l, "\alpha"'] \\
B \arrow[u, "H"] & \Sigma^2 B \arrow[l, "\beta"] \arrow[u, "\Sigma^2 H"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\Sigma^{X\times V} & \Sigma^{X\times U} \arrow[l, "J^X"'] \\
\Sigma^{Y\times V} \arrow[u, "H^V"] & \Sigma^{Y\times U} \arrow[l, "J^Y"] \arrow[u, "H^U"']
\end{tikzcd}$$

(b) A morphism $H : \Sigma^Y \to \Sigma^X$ is called ***central*** if for every morphism $J : \Sigma^U \to \Sigma^V$, the equation $J^Y \,;\, H^V = H^U \,;\, J^X$ holds, as in the square on the right.

We must be careful with the notation J^X, as it is ambiguous which way round the product is in the exponent.

3.5. Proposition. *If H is a homomorphism or central, and invertible in $\mathcal{C}$, then its inverse is also a homomorphism or central, respectively.* ∎

3.6. Lemma. *For any map $f : X \to Y$ in $\mathcal{C}$, the map $\Sigma^f : \Sigma^Y \to \Sigma^X$ is both a homomorphism and central.*

Proof. It is a homomorphism by naturality of η, and central by naturality of $J^{(-)}$, with respect to f. ∎

3.7. Lemma. *Every homomorphism $H : \Sigma^Y \to \Sigma^X$ is central.*

$$
\begin{tikzcd}
& \Sigma^{\Sigma^2 Y\times V} \arrow[rr, "\Sigma^{(\eta_Y\times V)}"] \arrow[dd, "\Sigma^{(\Sigma^H\times V)}"] & & \Sigma^{Y\times V} \arrow[dd, "H^V"] \\
\Sigma^{\Sigma^2 Y\times U} \arrow[ur, "J^{\Sigma^2 Y}"] \arrow[rr, shift left, "\Sigma^{\eta_Y\times U}"] \arrow[dd, "\Sigma^{(\Sigma^H\times U)}"'] & & \Sigma^{Y\times U} \arrow[ll, shift left, "\eta^U_{\Sigma Y}"] \arrow[ur, "J^Y"] \arrow[dd, "H^U"'] & ? \\
& \Sigma^{\Sigma^2 X\times V} \arrow[rr, "\Sigma^{(\eta_X\times V)}"] & & \Sigma^{X\times V} \\
\Sigma^{\Sigma^2 X\times U} \arrow[ur, "J^{\Sigma^2 X}"] \arrow[rr, "\Sigma^{(\eta_X\times U)}"] & & \Sigma^{X\times U} \arrow[ur, "J^X"'] &
\end{tikzcd}
$$

Proof. The front and back faces commute since H is a homomorphism. The left, bottom and top faces commute by naturality of $J^{(-)}$ with respect to Σ^H, η_X and η_Y. Using the split epi, the right face commutes, but this expresses centrality. ∎

3.8. Remark. To obtain the converse, we ought first to understand how monads provide a "higher order" account of infinitary algebraic theories [Lin69]. The infinitary theory corresponding to our monad has an operation-symbol J of arity U for each morphism $J : \Sigma^U \to \Sigma$ (for example, the additional lattice structure consists of morphisms $\wedge : \Sigma^{\mathbf{2}} \to \Sigma$ and $\bigvee : \Sigma^{\mathbb{N}} \to \Sigma$). Then the centrality square is the familiar rule for a homomorphism H to commute with the symbol J.

More generally, H commutes with $J : \Sigma^U \to \Sigma^V$ iff it is a homomorphism *parametrically* with respect to a V-indexed *family* of U-ary operation-symbols. (So, for example, a map $J : \Sigma^{\mathbf{3}} \to \Sigma^{\mathbf{2}}$ denotes a pair of ternary operations.) The next two results are examples of this idea, and we apply it to the lattice structure in the topological interpretation in Proposition 5.5.

3.9. Lemma. *$H : \Sigma^Y \to \Sigma^X$ is a homomorphism with respect to all constants $\sigma \in \Sigma$, i.e. $\sigma : \Sigma \vdash H(\lambda y.\, \sigma) = \lambda x.\, \sigma$, iff $\Sigma^{!_Y} ; H = \Sigma^{!_X}$.*

$$
\begin{tikzcd}
& \Sigma^{\Sigma} & & \Sigma^{\mathbf{0}} = \mathbf{1} \arrow[ll, "\widetilde{\mathsf{id}}"'] \\
\Sigma^X & \Sigma^{X\times\Sigma} & & \Sigma^{X\times\mathbf{0}} \arrow[ll, "\widetilde{\pi}_0"'] \\
\Sigma^Y \arrow[u, "H"] & \Sigma^{Y\times\Sigma} \arrow[u, "H^{\Sigma}"] & & \Sigma^{Y\times\mathbf{0}} \arrow[ll, "\widetilde{\pi}_0"'] \arrow[u, equal, "H^{\mathbf{0}}"']
\end{tikzcd}
$$

PROOF. $J = \widetilde{\mathsf{id}} : \mathbf{1} \to \Sigma^\Sigma$ denotes a Σ-indexed family of constants, for which the centrality square is as shown. ∎

The following proof is based on the idea that each $r \in TB$ corresponds to an operation-symbol of arity B that acts on A as $r_A : A^B \to A$ by $f \mapsto \alpha(Tfr)$. The rectangle says that K is a homomorphism for this operation-symbol, but it does so for all $r \in TB$ simultaneously by using the exponential $(-)^{TB}$.

Thielecke [Thi97a, Lemma 5.2.5] proves this result using his CPS λ-calculus, whilst Selinger [Sel01, Lemma 2.10] gives another categorical proof. They do so with surprisingly little comment, given that it is the *Completeness* Theorem corresponding to the easy Soundness Lemma 3.7.

3.10. THEOREM. *All central maps are homomorphisms.*

PROOF. Let $H : B = \Sigma^Y \to A = \Sigma^X$. Writing T for both the functor $\Sigma^{\Sigma^{(-)}}$ and its effect on internal hom-sets, consider the rectangle below, in which the left-hand square says that T preserves composition and the right-hand square that H is an Eilenberg–Moore homomorphism. To deduce the latter, it suffices to show that the rectangle commutes at $\mathsf{id} \in B^B$ (which T takes to $\mathsf{id} \in TB^{TB}$).

$$\begin{tikzcd}
B^B \arrow[r, "T"] \arrow[d, "H^B"'] \arrow[dr, phantom, "\text{functor}"] & TB^{TB} \arrow[r, "\beta^{TB}"] \arrow[d, "TH^{TB}"] \arrow[dr, phantom, "?"] & B^{TB} \arrow[d, "H^{TB}"] \\
A^B \arrow[r, "T"] & TA^{TB} \arrow[r, "\alpha^{TB}"] & A^{TB}
\end{tikzcd}$$

Re-expanding, the map along the bottom is $\Sigma^{X\times B} \to \Sigma^{\Sigma^2 X \times \Sigma^2 B} \to \Sigma^{X\times\Sigma^2 B}$ by

$$\theta \mapsto \lambda F\mathcal{G}.\, \mathcal{G}\big(\lambda b.\, F(\theta b)\big) \mapsto \lambda x\mathcal{G}.\, \mathcal{G}\big(\lambda b.\, \eta_X x(\theta b)\big) = \lambda x\mathcal{G}.\, \mathcal{G}(\lambda b.\, \theta b x),$$

which is $\eta^X_{\Sigma^B}$, and similarly the top map is $\eta^Y_{\Sigma^B}$. Thus the rectangle says that $H : \Sigma^Y \to \Sigma^X$ is central with respect to η_{Σ^B}, which was the hypothesis. ∎

3.11. NOTATION. We write $\mathcal{A}$ (or sometimes $\mathcal{A}_\mathcal{C}$) for the category of Eilenberg–Moore algebras and homomorphisms.

3.12. LEMMA. *For any object X, $(\Sigma^{\Sigma^X}, \mu_X)$ is the free algebra on X. In particular, $(\Sigma, \Sigma^{\eta_1})$ is the initial algebra.* ∎

3.13. DEFINITION. The full subcategory of $\mathcal{A}$ consisting of free algebras is known as the ***Kleisli category*** for the monad.

As Σ^{Σ^X} is free on X, the name of this object in the traditional presentation of the Kleisli category is abbreviated to X, and the homomorphism $X \to_K Y$ (*i.e.* $\Sigma^{\Sigma^X} \to \Sigma^{\Sigma^Y}$)

is named by the ordinary map $f : X \to \Sigma^{\Sigma^Y}$. This presentation is complicated by the fact that the identity on X is named by η_X and the composite of $f : X \to_K Y$ and $g : Y \to_K Z$ by $f \,;\, \Sigma^2 g \,;\, \mu_Z$.

Using the double exponential transpose (Proposition 2.11), this homomorphism is more simply written as an arbitrary map $F : \Sigma^Y \to \Sigma^X$, with the usual identity and composition.

The (opposites of the) categories composed of morphisms $F : \Sigma^Y \to \Sigma^X$, in the cases where F is an arbitrary $\mathcal{C}$-map (as for the Kleisli category), or required to be a homomorphism, will be developed in Section 6.

4. Sobriety and monadicity

Our new notion of sobriety, expressed in terms of the λ-calculus rather than lattice theory, is a weaker form of the fundamental idea of the abstract Stone duality programme.

4.1. Definition. When the category $\mathcal{C}$ of types of values is dual to its category $\mathcal{A}$ of algebras of observations, we say that $(\mathcal{C}, \Sigma)$ is ***monadic***. More precisely, the comparison functor $\mathcal{C}^{\mathsf{op}} \to \mathcal{A}$ defined by Lemmas 3.3 and 3.6,

$$
\begin{tikzcd}[row sep=huge, column sep=huge]
\mathcal{C}^{\mathsf{op}} \arrow[rr, "X \mapsto (\Sigma^X, \Sigma^{\eta_X})"] \arrow[d, shift left, "\Sigma^{(-)}"] \arrow[d, phantom, "\dashv"] & & \mathcal{A} \arrow[d, shift left, "(A,\alpha) \mapsto A"] \arrow[d, phantom, "\dashv"] \\
\mathcal{C} \arrow[u, shift left, "\Sigma^{(-)}"] \arrow[rr, equal] & & \mathcal{C} \arrow[u, shift left, "X \mapsto (\Sigma^{\Sigma^X},\ \mu_X)"]
\end{tikzcd}
$$

(which commutes both with the left adjoints and with the right adjoints) is to be an equivalence of categories, *i.e.* full, faithful and essentially surjective.

4.2. Remark. It is possible to characterise several weaker conditions than categorical equivalence, both in terms of properties of the objects of $\mathcal{C}$, and using generalised "mono" requirements on η_X. In particular, the functor $\mathcal{C}^{\mathsf{op}} \to \mathcal{A}$ is faithful iff all objects are "T_0" (*cf.* Remark 1.13), and also reflects invertibility iff they are replete [Hyl91, Tay91]. Another way to say this is that each η_X is mono or extremal mono, and a third is that Σ is a weak or strong cogenerator.

For example, $\mathbb{N}$ with primitive recursion is T_0 so long as the calculus is consistent, but repleteness and sobriety are equivalent to general recursion (Sections 9–10).

In this paper we are interested in the situation where the functor is full and faithful, *i.e.* that *all* homomorphisms are given uniquely by Lemma 3.6. We shall show that the corresponding property of the objects is sobriety, and that of η_X is that it be the equaliser of a certain diagram.

4.3. Lemma. *Let (A, α) be an algebra, Γ any object and $H : A \to \Sigma^\Gamma$ any map in $\mathcal{C}$. Then H is a homomorphism iff its double exponential transpose $P : \Gamma \to \Sigma^A$ (Proposition 2.11) has equal composites*

$$\begin{tikzcd} \Gamma \arrow[r, "P"] & \Sigma^A \arrow[r, shift left, "\Sigma^\alpha"] \arrow[r, shift right, "\eta_{\Sigma A}"'] & \Sigma^3 A. \end{tikzcd}$$

Proof. We have $H = \eta_A;\Sigma^P$ and $P = \eta_\Gamma;\Sigma^H$. $[\Rightarrow]$ $P;\Sigma^\alpha = \eta_\Gamma;\Sigma^H;\Sigma^\alpha = \eta_\Gamma;\Sigma^2\eta_\Gamma;\Sigma^3 H = \eta_\Gamma ; \eta_{\Sigma^2\Gamma} ; \Sigma^3 H = \eta_\Gamma ; \Sigma^H ; \eta_{\Sigma A} = P ; \eta_{\Sigma A}$. $[\Leftarrow]$ $\alpha ; H = \alpha ; \eta_A ; \Sigma^P = \eta_{\Sigma^2 A} ; \Sigma^2\alpha ; \Sigma^P = \eta_{\Sigma^2 A} ; \Sigma\eta_{\Sigma A} ; \Sigma^P = \Sigma^P = \Sigma^2\eta_A ; \Sigma\eta_{\Sigma A} ; \Sigma^P = \Sigma^2\eta_A ; \Sigma^3 P ; \Sigma\eta_\Gamma = \Sigma^2 H ; \Sigma\eta_\Gamma$. ∎

4.4. Corollary. *The (global) elements of the equaliser are the those functions $A \to \Sigma$ that are homomorphisms.* ∎

4.5. Definition. Such a map P is called ***prime***. (We strike through the history of uses of this word, such as in Definition 1.7 and Corollary 5.8. In particular, although the case $X = \mathbb{N}$ will turn out to be the most important one, we are *not* just talking about the numbers 2, 3, 5, 7, 11, ...!) As we always have $A = \Sigma^X$ in this paper, we usually write P as a term $\Gamma \vdash P : \Sigma^{\Sigma^X}$.

4.6. Lemma. *In Lemma 4.3, η_X is the prime corresponding to the homomorphism $\mathsf{id} : \Sigma^X \to \Sigma^X$. If $P : \Gamma \to \Sigma^A$ is prime and $J : B \to A$ a homomorphism then $P ; \Sigma^J$ is also prime. In particular, composition with $\Sigma^2 f$ preserves primes.* ∎

4.7. Definition. We say that an object $X \in \mathsf{ob}\,\mathcal{C}$ is ***sober*** if the diagram

$$\begin{tikzcd} X \arrow[r, tail, "\eta_X"] & \Sigma^{\Sigma^X} \arrow[r, shift left, "\Sigma^2(\eta_X)"] \arrow[r, shift right, "\eta_{\Sigma^2 X}"'] & \Sigma^4(X) \end{tikzcd}$$

is an equaliser in $\mathcal{C}$, or, equivalently, that the naturality square

$$\begin{tikzcd} X \arrow[r, tail, "\eta_X"] \arrow[d, tail, "\eta_X"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Sigma^{\Sigma^X} \arrow[d, "\Sigma^2(\eta_X)"] \\ \Sigma^{\Sigma^X} \arrow[r, tail, "\eta_{\Sigma^2 X}"] & \Sigma^4(X) \end{tikzcd}$$

for η with respect to η_X is a pullback.

4.8. Remark. We have only said that the *existing* objects can be *expressed* as equalisers, not that general equalisers can be *formed*. In fact, this equaliser is of the special form described below, which Jon Beck exploited to characterise monadic adjunctions [Mac71, Section VI 7], [BW85, Section 3.3], [Tay99, Section 7.5]. The category will be extended to include such equalisers, so we recover a space $\mathsf{pts}(A, \alpha)$ from any algebra, in [B].

Notice the double role of Σ here, as both a space and an algebra. Peter Johnstone has given an account of numerous well known dualities [Joh82, Section VI 4] based the idea that Σ is a ***schizophrenic object***. (This word was first used by Harold Simmons, in a draft of [Sim82], but removed from the published version.)

Moggi [Mog88] called sobriety the *equalizing requirement*, but did not make essential use of it in the development of his computational monads.

Applegate and Tierney [Eck69, p175] and Barr and Wells [BW85, Theorem 3.9.9] attribute these results for general monads to Jon Beck. See also [KP93] for a deeper study of this situation.

4.9. Proposition. *Any power, Σ^U, is sober.*

$$\begin{tikzcd}
\Sigma^U \arrow[r, shift left, "\eta_{\Sigma^U}"] & \Sigma^3 U \arrow[l, shift left, dotted, "\Sigma^{\eta_U}"] \arrow[r, shift left=2, "\eta_{\Sigma^3 U}"] \arrow[r, "\Sigma^2 \eta_{\Sigma^U}" description] & \Sigma^5 U \arrow[l, shift left=2, dotted, "\Sigma^3 \eta_U"]
\end{tikzcd}$$

Proof. This is a *split* equaliser: the dotted maps satisfy

$$\begin{array}{rclcrcl}
\eta_{\Sigma^U} \,;\, \Sigma^{\eta_U} & = & \mathsf{id}_{\Sigma^U} & \qquad & \Sigma^{\eta_U} \,;\, \eta_{\Sigma^U} & = & \eta_{\Sigma^3 U} \,;\, \Sigma^3 \eta_U \\
\Sigma^2 \eta_{\Sigma^U} \,;\, \Sigma^3 \eta_U & = & \mathsf{id}_{\Sigma^3 U} & & \eta_{\Sigma^U} \,;\, \eta_{\Sigma^3 U} & = & \eta_{\Sigma^U} \,;\, \Sigma^2 \eta_{\Sigma^U}
\end{array}$$

by Lemma 2.10, the equations on the right being naturality of η with respect to Σ^{η_U} and η_{Σ^U}. Hence if $P : \Gamma \to \Sigma^3 U$ has equal composites then $P = P ; \Sigma^{\eta_U} ; \eta_{\Sigma^U}$, and the mediator is $P ; \Sigma^{\eta_U}$. ∎

4.10. Theorem. *The functor $\Sigma^{(-)} : \mathcal{C}^{\mathsf{op}} \to \mathcal{A}$ given in Definition 4.1 is full and faithful iff all objects are sober.*

Proof. [$\Rightarrow$] We use $P : \Gamma \to \Sigma^2 X$ to test the equaliser. By Lemma 4.3, its double transpose $H : \Sigma^X \to \Sigma^\Gamma$ is a homomorphism, so by hypothesis $H = \Sigma^f = \eta_{\Sigma^X} ; \Sigma^P$ for some unique $f : \Gamma \to X$, and this mediates to the equaliser.

$$\begin{tikzcd}
\Sigma^\Gamma & \Sigma^3 \Gamma \arrow[l, "\Sigma^{\eta_\Gamma}"'] \\
\Sigma^X \arrow[u, "H"] & \Sigma^3 X \arrow[l, "\Sigma^{\eta_X}"] \arrow[u, "\Sigma^2 H"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\Gamma \arrow[r, "\eta_\Gamma"] \arrow[d, dotted, "f"'] & \Sigma^2 \Gamma \arrow[r, shift left, "\eta_{\Sigma^2 \Gamma}"] \arrow[r, shift right, "\Sigma^2 \eta_\Gamma"'] \arrow[d, "\Sigma^H"'] & \Sigma^4 \Gamma \arrow[d, "\Sigma^3 H"] \\
X \arrow[r, "\eta_X"] & \Sigma^2 X \arrow[r, shift left, "\eta_{\Sigma^2 X}"] \arrow[r, shift right, "\Sigma^2 \eta_X"'] & \Sigma^4 X
\end{tikzcd}$$

[$\Leftarrow$] Let $H : \Sigma^X \to \Sigma^\Gamma$ be a homomorphism, so the diagram on the left above commutes, as do the parallel squares on the right, the lower one by naturality of Σ^η with respect to H. Since X is the equaliser, there is a unique mediator $f : \Gamma \to X$, and we then have $H = \eta_{\Sigma^X} ; \Sigma^P = \eta_{\Sigma^X} ; \Sigma\eta_X ; \Sigma^f = \Sigma^f$. ∎

4.11. Remark. Translating Definition 3.4(a) into the λ-calculus, the property of being a homomorphism $H : \Sigma^X \to \Sigma^U$ can be expressed in a finitary way as an equation between λ-expressions,

$$\mathcal{F} : \Sigma^3 X \vdash \big(\lambda u.\, \mathcal{F}(\lambda\phi.\, H\phi u)\big) = H\big(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)\big),$$

the two sides of which differ only in the position of H.

The double exponential transpose P of H is obtained in the λ-calculus simply by switching the arguments ϕ and u (*cf.* Remark 1.11). Hence $\vdash P : U \to \Sigma^{\Sigma^X}$ is prime iff

$$u : U,\ \mathcal{F} : \Sigma^3 X \vdash \mathcal{F}(Pu) = Pu\big(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)\big).$$

Replacing the argument u of P by a context Γ of free variables, $\Gamma \vdash P : \Sigma^{\Sigma^X}$ is prime iff

$$\Gamma,\ \mathcal{F} : \Sigma^3 X \ \vdash\ \mathcal{F}P \ =\ P\big(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)\big)$$

or $\mathcal{F}P = P(\eta_X ; \mathcal{F})$. This is the equation in Lemma 4.3, with $A = \Sigma^X$, applied to $\mathcal{F}$.

4.12. Corollary. *The type X is sober iff for every prime $\Gamma \vdash P : \Sigma^2 X$ there is a unique term $\Gamma \vdash \mathsf{focus}\, P : X$ such that*

$$\Gamma,\ \phi : \Sigma^X \ \vdash\ \phi(\mathsf{focus}\, P) \ =\ P\phi. \qquad ∎$$

Hence the side-condition on the introduction rule for $\mathsf{focus}\, P$ in Remark 2.12 is that P be prime. Indeed, since $\phi \mapsto \phi x$ is itself a homomorphism (for fixed x), this equation is only meaningful in a denotational reading of the calculus when P is prime. (On the other hand Thielecke's force operation has this as a β-rule, with no side condition, but specifies a particular order of evaluation.)

4.13. Remark. So far, we have used none of the special structure on Σ in Remarks 2.4ff. We have merely used the restricted λ-calculus to discuss what it means for the other objects of the category to be sober with respect to it. In Sections 6–8 we shall show how to enforce this kind of sobriety on them.

If $P = \eta_X(x)$ then the right hand side of the primality equation easily reduces to the left. Otherwise, since $\mathcal{F}$ is a *variable*, the left hand side is head-normal, and so cannot be reduced without using an axiom such as the Euclidean principle (Remark 2.4), as we shall do in Proposition 10.6.

The introduction of subspaces [B] also extends the applicability of the equation, by allowing it to be proved *under hypotheses*, whilst using the continuity axiom (Remark 2.6) it is sufficient to verify that P or H preserves the lattice connectives. In other words, the *mathematical* investigations to follow serve to show that the required denotational results are correctly obtained by *programming* with computational effects.

4.14. Remark. In his work on continuations, Hayo Thielecke uses R for our Σ and interprets it as the ***answer type***. This is the type of a sub-program that is called like a function, but, since it passes control by calling another continuation, never returns "normally" — so the type of the answer is irrelevant. Thielecke stresses that R therefore has no particular properties or structure of its own.

In the next section, we shall show that the Sierpiński space in topology behaves categorically in the way that we have discussed, but it does carry additional lattice-theoretic structure.

Even though a function or procedure of type `void` never returns a "numerical" result — and may never `return` at all — it does have the undisguisable behaviour of termination or non-termination. Indeed, we argued in Remark 1.2 that termination is the ultimate desideratum, and that therefore the type of observations should also carry the lattice structure. Proposition 10.6, which I feel does impact rather directly on computation, makes use of both this structure and the Euclidean principle.

Thielecke's point of view is supported by the fact that the class of objects that are deemed sober depends rather weakly on the choice of object Σ: in classical domain theory, any non-trivial Scott domain would yield the same class.

4.15. Remark. Although it belongs in general topology, sobriety was first used by the Grothendieck school in algebraic geometry [AGV64, IV 4.2.1] [GD71, 0.2.1.1] [Hak72, II 2.4]. They exploited *sheaf theory*, in particular the functoriality of constructions with respect to the lattice of open subsets, the points being secondary.

An *algebraic variety* (the set of solutions of a system of polynomial equations) is closed in the Euclidean topology, but there is a coarser *Zariski topology* in which they are *defined* to be closed. When the polynomials do not factorise, the closed set is not the union of non-trivial closed subsets, and is said to be *irreducible*. A space is sober (classically) iff every irreducible closed set is the closure of a unique point, known in geometry as the *generic* point of the variety. Such generic points, which do not exist in the classical Euclidean topology, had long been a feature of geometrical reasoning, in particular in the work of Veronese (*c.* 1900), but it was Grothendieck who made their use rigorous.

5. Topology revisited

In this section we show how the abstract categorical and symbolic structures that we have introduced are equivalent to the traditional notions in general topology that we mentioned in Section 1. In fact, all that we need to do is to *re-interpret* lattice-theoretic work that was done in the 1970s. On this occasion our treatment will be entirely classical, making full use of the axiom of choice and excluded middle; for a more careful intuitionistic account see [B, C].

If you are not familiar with locally compact topological spaces, you may consider instead your favourite category of algebraic (or continuous) predomains, which are all sober. The discrete space $\mathbb{N}$ is also needed, besides domains with $\perp$. The results of this section are only used as motivation, so you can in fact omit it altogether.

Alternatively, the construction may be performed with arbitrary dcpos, although it adds extra points to those that are not sober. Peter Johnstone gave an example of such a non-sober dcpo [Joh82, Exercise II 1.9], as part of the philosophical argument against *point*-set topology. We shall not need this, as the localic view is already deeply embedded in our approach. In fact, when we construct new spaces in [B], they will be carved out as subspaces of lattices (*cf.* [Sco72]) not glued together from points.

5.1. Remark. The classical ***Sierpiński space*** Σ has two points: $\top$ is open and $\bot$ is closed. So altogether there are three open sets: $\emptyset$, $\{\top\}$ and Σ.

This space has the (universal) property that, for any open subset U of any space X, there is a unique continuous function $f : X \to \Sigma$ such that the inverse image $f^*\top$ is U. Indeed, f takes the points of U to $\top$, and those of its closed complement to $\bot$.

$$\begin{tikzcd} U \arrow[r] \arrow[d, hook] \arrow[dr, phantom, very near start, "\lrcorner"] & \{\top\} \arrow[d] \\ X \arrow[r, "f"] & \Sigma \end{tikzcd}$$

This is the same as the defining property of the ***subobject classifier*** Ω in a topos, except that there $U \subset X$ can be *any* subobject. We shall discuss sobriety for sets, discrete spaces and objects of a topos in Section 9.

Hence open subsets of X correspond bijectively to maps $X \to \Sigma$, and so to points of the exponential Σ^X. In other words, the space Σ^X is the lattice of open subsets of X, equipped with some topology.

5.2. Remark. Finite intersections and arbitrary unions of open subsets give rise to internal lattice structure on Σ, written $\wedge : \Sigma \times \Sigma \to \Sigma$ and $\bigvee : \Sigma^U \to \Sigma$. Besides the infinite distributive law, conjunction also satisfies the Euclidean principle (Remark 2.4). Whilst this is vacuous classically, it and its lattice dual (which says that $\bot$ classifies closed subsets) capture remarkably much of the flavour of locale theory [C, D], before we need to invoke the continuity axiom (Remark 2.6), though of course that is also valid in topology.

5.3. Remark. To determine the topology on the space Σ^X, consider the map $\mathsf{ev} : \Sigma^X \times X \to \Sigma$. For this to be continuous, Ralph Fox showed that the space X must be locally compact, and Σ^X must have the compact–open topology [Fox45], which is the same as the Scott topology when we only consider Σ and not more general target spaces. The categorical analysis is due to John Isbell [Isb75].

Local compactness is a very familiar notion for Hausdorff spaces, but there are messy subtleties to its definition for non-Hausdorff spaces [HM81]. However, so long as we only consider spaces that are sober in the standard topological sense, things are not too difficult:

For any point x and open subset $x \in U \subset X$, there must be a compact subset K and another open subset V with $x \in V \subset K \subset U$. The "open rectangle" around

$(U, x) \in \mathsf{ev}^{-1}\{\top\} \subset \Sigma^X \times X$ that we need for continuity of ev is then

$$\{W \in \Sigma^X \mid K \subset W\} \times V \;\subset\; \Sigma^X \times X.$$

In the jargon, X ***has a base of compact neighbourhoods***, *cf.* Bourbaki's usage in Remark 1.8.

All of this is much prettier in terms of the open sets: the topology Σ^X is a ***distributive continuous lattice***, equipped with the Scott topology. Such a lattice is of course a frame, and the corresponding locale is called locally compact. Assuming the axiom of choice, the category **LKLoc** of locally compact locales is equivalent to the category **LKSp** of locally compact sober spaces [Joh82, Section VII 4.5].

5.4. Remark. Since Σ^X carries the Scott topology, a continuous function $\Sigma^Y \to \Sigma^X$ is a function between open set lattices that preserves directed unions. Such a function is called ***Scott-continuous***. In particular, it preserves the order that is induced by the lattice structure [Sco72].

Besides frame homomorphisms themselves, functions like this between frames do arise in general topology. For example, a space K is compact iff $\Sigma^{!_K} : \Sigma \to \Sigma^K$ has a *Scott-continuous* right adjoint, $\bigwedge : \Sigma^K \to \Sigma$. Unfortunately, monotone functions between frames that need not preserve directed joins are also used in general topology, and these present the main difficulty that abstract Stone duality faces in re-formulating the subject [D].

5.5. Proposition. *Let $H : \Sigma^X \to \Sigma^U$ be a Scott-continuous function between the topologies of locally compact spaces. If H is central (Definition 3.4(b)) then it preserves finite meets and arbitrary joins.*

$$\begin{tikzcd} \Sigma & \Sigma^{\mathbf{2}} \arrow[l, "\wedge"'] \end{tikzcd} \qquad \begin{tikzcd} \Sigma & \Sigma^{\mathbb{N}} \arrow[l, "\bigvee"'] \end{tikzcd}$$

$$\begin{tikzcd} \Sigma^U \\ \Sigma^X \arrow[u, "H"] \end{tikzcd} \qquad \begin{tikzcd} \Sigma^U & (\Sigma^U)^{\mathbf{2}} \arrow[l, "\wedge^U"'] \\ \Sigma^X \arrow[u, "H"] & (\Sigma^X)^{\mathbf{2}} \arrow[l, "\wedge^X"'] \arrow[u, "H^{\mathbf{2}}"'] \end{tikzcd} \qquad \begin{tikzcd} \Sigma^U & (\Sigma^U)^{\mathbb{N}} \arrow[l, "\bigvee^U"'] \\ \Sigma^X \arrow[u, "H"] & (\Sigma^X)^{\mathbb{N}} \arrow[l, "\bigvee^X"'] \arrow[u, "H^{\mathbb{N}}"'] \end{tikzcd}$$

Proof. Lemma 3.9 dealt with the constants ($\top$ and $\bot$), so consider $J = \wedge : \Sigma^{\mathbf{2}} \to \Sigma$ and $\bigvee : \Sigma^{\mathbb{N}} \to \Sigma$. ∎

5.6. Remark. We also have $\bigvee : \Sigma^U \to \Sigma$ for any space U, together with the associated distributive law. In the case of $U = \mathbb{N}$, we write $\exists$ for $\bigvee$, and distributivity is known as the ***Frobenius law***,

$$\psi \wedge \exists n.\, \phi(n) \;=\; \exists n.\, \psi \wedge \phi(n).$$

We have $\bigwedge : \Sigma^K \to \Sigma$ only when K is compact; in particular, it would be $\forall$ for $K = \mathbb{N}$, but ($\mathbb{N}$ is not a compact space and) $\forall_{\mathbb{N}}$ is not computable (*cf.* Definition 1.3). Indeed, in a constructive setting, $\bigvee : \Sigma^U \to \Sigma$ only exists for *certain* spaces U, which are called overt Section[1] C 8. Overtness is analogous to recursive enumerability, *cf.* Remark 9.12 and Lemma 10.2.

We are now ready to show how our new λ-calculus formulation in Sections 3–4 captures the hitherto lattice-theoretic ideas of continuous functions and sober spaces. A proof entirely within abstract Stone duality (including the continuity axiom) that preserving the lattice operations suffices will be given in [E].

5.7. Theorem. *Let U and X be locally compact sober spaces and $H : \Sigma^X \to \Sigma^U$ a Scott-continuous function between their topologies. Then the following are equivalent:*
(a) *$H = \Sigma^f$ for some unique continuous function $f : U \to X$;*
(b) *H preserves finite meets and joins ($\top$, $\bot$, $\wedge$ and $\vee$);*
(c) *H is a frame homomorphism, i.e. it preserves $\top$, $\wedge$ and $\bigvee$;*
(d) *H is central (Definition 3.4(b));*
(e) *H is an Eilenberg–Moore homomorphism (Definition 3.4(a));*
(f) *H satisfies the equation in Remark 4.11.*

Proof. We have just shown that central maps are frame homomorphisms.

Since X is sober in the topological sense, all frame homomorphisms $\Sigma^X \to \Sigma^U$ are of the form Σ^f for some unique $f : U \to X$ (we take this as the topological definition of sobriety). But all Σ^f are Eilenberg–Moore homomorphisms. ∎

5.8. Corollary. *The following are equivalent for $P : \mathbf{1} \to \Sigma^{\Sigma^X}$:*
(a) *$P \subset \Sigma^X$ is the set $\eta_X(x)$ of open neighbourhoods of some unique point $x \in X$;*
(b) *P is a coprime filter*
(c) *its complement, $\Sigma^X \setminus P$, is a prime ideal;*
(d) *P is a completely coprime filter;*
(e) *P is a point of the equaliser in Lemma 4.3;*
(f) *P satisfies the equation in Remark 4.11.*
Similarly, for a continuous function $P : U \to \Sigma^{\Sigma^X}$, the same equivalent conditions hold for $P(u)$ for each point $u \in U$. ∎

5.9. Remark. Our primes are therefore what Johnstone calls the "points" of the locale Σ^X, so sobriety for **LKSp** in our sense agrees with his [Joh82, Section II 1.6]. As a topological space, the equaliser is the set U of primes, equipped with the sparsest locally compact topology such that $U \to \Sigma^2 X$ is continuous, and the hom-frame $\mathcal{C}(X, \Sigma)$ provides this topology.

[1] My paper *Geometric and Higher Order Logic* is cited as if it were "Chapter" C of a book.

5.10. Remark. We have a *theorem* in the straightforward sense for **LKSp** that says that, *given* a Scott-continuous map $H : \Sigma^X \to \Sigma^U$ between the open-set lattices of *given* locally compact spaces, H is a homomorphism of frames *if and only if* it is a homomorphism in the sense of our monad.

By contrast, the notions of sobriety expressed in terms of lattice theory and the λ-calculus agree only in *intuition*. We are only able to bring these two mathematical systems together in a setting where topological sobriety has already been assumed. If you are skeptical of the mathematical status of the argument, consider the analogous question in the relationship between locales and Bourbakian spaces: at what point in the axiomatisation of locales do we make the assumption that renders them all sober? Even then, these two categories only agree on their products on the same subcategory as ours, namely locally compact spaces. In summary, the concordance of several approaches (along with [E], and models of synthetic domain theory [Tay91]) makes us confident that the notion of locally compact space is a good one, but not so sure how it ought to be generalised.

5.11. Remark. The types of the restricted λ-calculus, even with the additional lattice and recursive structure, form a very impoverished category of spaces. Identifying them with their interpretations in **LKSp**, they amount merely to (some of) the algebraic lattices that Dana Scott used in the earliest versions of his denotational semantics [Sco76], and include no spaces at all (apart from $\mathbf{1}$ and $\mathbb{N}$) that would be recognisable to a geometric topologist.

The monadic property populates the category of spaces with *subspaces* of the types of the restricted λ-calculus. We show how to do this in terms of both abstract category theory and as an extension of the λ-calculus similar to the axiom of comprehension in [B], which also proves the next result intuitionistically for locales.

Although we only intended to consider sobriety and not monadicity in this paper, we actually already have enough tools to characterise the algebras for the monad classically in lattice-theoretic terms. Two more proofs appear in [B].

5.12. Theorem. **LKSp** *is monadic.*

Proof. As all of the spaces are sober, the functor in Definition 4.1 is full and faithful. It remains to show that *every* Eilenberg–Moore algebra (A, α) is of the form $(\Sigma^X, \Sigma^{\eta_X})$ for some locally compact space X. But, as A is a retract of a power of Σ, it must be a continuous lattice equipped with the Scott topology, and must in fact also be distributive, so $A \cong \Sigma^X$ for some locally compact sober space X.

However, we still need to show that the Eilenberg–Moore structure $\alpha : \Sigma^2 A \to A$ is uniquely determined by the order on A, and is therefore Σ^{η_X} as in Lemma 3.3.

For this, we must determine αF for each element $F \in \Sigma^2 A$; such F defines a Scott-open subset of the lattice Σ^A. It can be expressed as a union of Scott-open *filters* in this lattice, *i.e.* these filters form a base for the Scott topology [Joh82, Lemma VII 2.5] [GHK+80, Section I.3]. Since α must preserve unions, it suffices to define αF when F is a Scott-open filter.

Now each Scott-open filter F itself corresponds to compact saturated subspace $K \subset A$ [HM81, Theorem 2.16], *cf.* Remark 1.10. For a lattice A with its Scott topology, a "saturated" subspace is simply an upper set. Since α must be monotone and satisfy $\eta_A\,;\alpha = \mathsf{id}_A$, we have $\alpha(F) \leq a$ for all $a \in K$. For the same reason, if $b \in A$ satisfies $\forall a.\, a \in K \Rightarrow b \leq a$ then $b \leq \alpha(K)$. Hence $\alpha(F) = \bigwedge K \in A$. Thus the effect of α on all elements of $\Sigma^2 A$ has been fixed, as a join of meets. ■

Setting aside the discussion of more general spaces, what we learn from this is that, when all objects are sober, there are "second class" maps between objects. We shall see in the next section that this phenomenon arises for abstract reasons, and that the potential confusion over (Scott-) "continuous maps between frames" is not an accident.

6. Enforcing sobriety

Now we turn from the analysis to the synthesis of categories that have all objects sober. So our primary interest shifts from locales to the restricted λ-calculus in Remark 2.1.

Since we handle continuous functions $f : X \to Y$ in terms of the corresponding inverse image maps Σ^f, it is natural to work in a category in which there are both "first class" maps $f : X \to Y$ (given concretely by homomorphisms $\Sigma^f : \Sigma^Y \to \Sigma^X$) and "second class" maps $\widehat{F} : X \longrightarrow\!\!\!\!\times\; Y$ that are specified by *any* $F : \Sigma^Y \to \Sigma^X$.

These second class maps — ordinary functions rather than homomorphisms between algebras — are just what is needed to talk about U-split (co)equalisers as in Beck's theorem (*cf.* Proposition 4.9 and [B]). Even in more traditional subjects such as group and ring theory, we do indeed sometimes need to talk about *functions* between algebras that are not necessarily homomorphisms.

The practical reason for according these maps a public definition is that the product functor is defined for them (Proposition 6.5), and this will be crucial for constructing the product of formal Σ-split subspaces in [B]. After I had hesitated on this point myself, it was seeing the work of Hayo Thielecke [Thi97b] and Carsten Führmann [Füh99] on continuations that persuaded me that this is the best technical setting, and this section essentially describes their construction.

As to the first class maps, the whole point of sobriety is that they consist not only of $f : X \to Y$ in $\mathcal{C}$, but other maps suitably defined in terms of the topology.

6.1. Definition. The categories $\mathsf{H}\mathcal{C}$ and $\mathsf{S}\mathcal{C}$ both have the same objects as $\mathcal{C}$, but

(a) the (second class) morphisms $\widehat{F} : X \longrightarrow\!\!\!\!\times\; Y$ in $\mathsf{H}\mathcal{C}$ are (any) $\mathcal{C}$-morphisms $F : \Sigma^Y \to \Sigma^X$, *cf.* Remark 3.13, and

(b) the (first class) morphisms $\widehat{H} : X \longrightarrow Y$ in $\mathsf{S}\mathcal{C}$ are $\mathcal{C}$-morphisms $H : \Sigma^Y \to \Sigma^X$ that are homomorphisms (Definition 3.4):

$$
\begin{tikzcd}
\Sigma^X & & \Sigma^3 X \arrow[ll, "\Sigma^{\eta_X}"'] \\
\Sigma^Y \arrow[u, "H"] & & \Sigma^3 Y \arrow[ll, "\Sigma^{\eta_Y}"] \arrow[u, "\Sigma^2 H"']
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[rr, "\eta_X"] \arrow[d, "\widehat{H}"'] & & \Sigma^2 X \arrow[d, "\Sigma^H"] \\
Y \arrow[rr, "\eta_Y"'] & & \Sigma^2 Y
\end{tikzcd}
$$

Thielecke and Führmann call these ***thunkable*** morphisms, since they write thunk_X for η_X considered as an $\mathsf{H}\mathcal{C}$-map. It follows immediately (given Theorem 3.10) that $\eta_X : X \to \Sigma^{\Sigma^X}$ is natural in $\mathsf{S}\mathcal{C}$ but not $\mathsf{H}\mathcal{C}$.

Identity and composition are inherited in the obvious way from $\mathcal{C}$, though contravariantly, which is why we need the $\widehat{F}$ notation (which [Sel01, Section 2.9] also uses).

6.2. Remark. By Lemma 3.6, for $f : X \to Y$ in $\mathcal{C}$, the map $H = \Sigma^f : \Sigma^Y \to \Sigma^X$ is a homomorphism, so $\widehat{H} : X \to Y$ is in $\mathsf{S}\mathcal{C}$. We shall just write this as $f : X \to Y$ instead of $\widehat{\Sigma^f} : X \to Y$, but beware that, in general, different $\mathcal{C}$-maps can become equal $\mathsf{S}\mathcal{C}$-maps with the same names.

By Remark 4.2, this functor $\mathcal{C} \to \mathsf{S}\mathcal{C}$ is faithful iff every object of $\mathcal{C}$ is T_0, and it also reflects invertibility if every object is replete. Theorem 4.10 said that it is also full iff every object of $\mathcal{C}$ is sober; as $\mathsf{S}\mathcal{C}$ and $\mathcal{C}$ have the same objects, they are then isomorphic categories.

There are, of course, many more morphisms in $\mathsf{H}\mathcal{C}$ than in $\mathcal{C}$, but one (family) in particular generates the rest. We shall see that the new second class *morphism* $\mathsf{force} : \Sigma^2 X \longrightarrow\!\!\!\!\times\; X$ objectifies the *operation* $P \mapsto \mathsf{focus}\, P$ that is only defined when $P : \Sigma^2 X$ is prime. Thielecke and Führmann apply their β-rule for force without restriction, producing computational effects, whilst our side-condition on focus gives it its denotational or topological meaning.

6.3. Definition. $\mathsf{force}_X = \widehat{\eta_{\Sigma^X}} : \Sigma^{\Sigma^X} \longrightarrow\!\!\!\!\times\; X$ is a natural transformation in the category $\mathsf{H}\mathcal{C}$, and satisfies $\eta_X \,;\mathsf{force}_X = \mathsf{id}_X$ or $\mathsf{force}(\mathsf{thunk}\, x) = x$.

Proof. This is Lemma 2.10 again. $\mathsf{force}_{(-)}$ in $\mathsf{H}\mathcal{C}$ is $\eta_{\Sigma^{(-)}}$ in $\mathcal{C}$, which is natural (in $\mathcal{C}$) with respect to all maps $F : \Sigma^Y \to \Sigma^X$, so $\mathsf{force}_{(-)}$ is natural (in $\mathsf{H}\mathcal{C}$) with respect to $\widehat{F} : X \longrightarrow\!\!\!\!\times\; Y$. The other equation is $\eta_{\Sigma^X} \,; \Sigma^{\eta_X} = \mathsf{id}_X$. ∎

6.4. Corollary. *The $\mathcal{C}$-map $\eta_X : X \to \Sigma^2 X$ is mono in both $\mathsf{H}\mathcal{C}$ and $\mathsf{S}\mathcal{C}$.*

Proof. It is split mono in $\mathsf{H}\mathcal{C}$, and remains mono in $\mathsf{S}\mathcal{C}$ because there are fewer pairs of incoming maps to test the definition of mono. ∎

6.5. Proposition. *For each object X, the product $X \times -$ in $\mathcal{C}$ extends to an endofunctor on $\mathsf{H}\mathcal{C}$. This construction is natural with respect to $\mathcal{C}$-maps $f : X \to Y$ (so $H = \Sigma^f$).*

$$
\begin{tikzcd}
\Sigma^{X\times U} & \Sigma^{X\times V} \arrow[l, "J^X"'] \\
\Sigma^{Y\times U} \arrow[u, "\Sigma^{f\times U}"] & \Sigma^{Y\times V} \arrow[l, "J^Y"'] \arrow[u, "\Sigma^{f\times V}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
X\times U \arrow[r, "X\times\widehat{J}"] \arrow[d, "f\times U"'] & X\times V \arrow[d, "f\times V"] \\
Y\times U \arrow[r, "Y\times\widehat{J}"] & Y\times V
\end{tikzcd}
$$

Proof. For $\widehat{J} : V \longrightarrow\!\!\!\!\!\times\, U$ in $\mathsf{H}\mathcal{C}$, *i.e.* $J : \Sigma^U \to \Sigma^V$ in $\mathcal{C}$, we write $X \times \widehat{J} : X \times V \longrightarrow\!\!\!\!\!\times\, X \times U$ for the $\mathcal{C}$-map $J^X : \Sigma^{X\times U} \to \Sigma^{X\times V}$. This construction preserves identities and composition because it is just the endofunctor $(-)^X$ defined on a subcategory of $\mathcal{C}$. It *extends* the product functor because, in the case of a first class map $g : V \to U$ (so $J = \Sigma^g : \Sigma^U \to \Sigma^V$), we have $X \times \widehat{J} = X \times \widehat{\Sigma^g} = \widehat{\Sigma^{X\times g}} = X \times g$, which is a first class map $X \times V \to X \times U$.

The construction is natural with respect to $f : X \to Y$ because $J^{(-)}$ is. ∎

6.6. Example. The existential quantifier $\exists^\Gamma_{\mathbb{N}}$ in the context Γ is obtained in this way, and its Beck–Chevalley condition with respect to the substitution or cut $f : \Gamma \to \Delta$ (Proposition C 8.1) is commutativity of the square (*cf.* Proposition 5.5):

$$
\begin{tikzcd}
\Sigma^{\Gamma} & \Sigma^{\Gamma\times\mathbb{N}} \arrow[l, "\exists^\Gamma_{\mathbb{N}}"'] \\
\Sigma^{\Delta} \arrow[u, "\Sigma^{f}"] & \Sigma^{\Delta\times\mathbb{N}} \arrow[l, "\exists^\Delta_{\mathbb{N}}"'] \arrow[u, "\Sigma^{f\times\mathbb{N}}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
\Gamma \arrow[r, "\Gamma\times\widehat{\exists}_{\mathbb{N}}"] \arrow[d, "f"'] & \Gamma\times\mathbb{N} \arrow[d, "f\times\mathbb{N}"] \\
\Delta \arrow[r, "\Delta\times\widehat{\exists}_{\mathbb{N}}"] & \Delta\times\mathbb{N}
\end{tikzcd}
$$

6.7. Example. $\times$ is not defined as a functor of *two* variables on $\mathsf{H}\mathcal{C}$, because the squares

$$
\begin{tikzcd}
\Sigma^{X\times U} & \Sigma^{X\times V} \arrow[l, "J^X"'] \\
\Sigma^{Y\times U} \arrow[u, "F^{U}"] \arrow[ur, phantom, "⊹"] & \Sigma^{Y\times V} \arrow[l, "J^Y"'] \arrow[u, "F^{V}"']
\end{tikzcd}
\qquad
\begin{tikzcd}
X\times U \arrow[r, "X\times\widehat{J}"] \arrow[d, "\widehat{F}\times U"'] \arrow[dr, phantom, "⊹"] & X\times V \arrow[d, "\widehat{F}\times V"] \\
Y\times U \arrow[r, "Y\times\widehat{J}"] & Y\times V
\end{tikzcd}
$$

do not necessarily commute (⊹ [FS90]). For example, take $\widehat{F} = \widehat{J} : \Sigma \longrightarrow\!\!\!\!\!\times\, \mathbf{0}$ where $F = J : \Sigma^{\mathbf{0}} = \mathbf{1} \to \Sigma^\Sigma$ is the element $\widetilde{\mathsf{id}} \in \Sigma^\Sigma$; then these two composites give the elements $\widetilde{\pi}_0, \widetilde{\pi}_1 \in \Sigma^{\Sigma\times\Sigma}$.

6.8. Remark. $\times$ is a ***premonoidal*** structure on $\mathsf{H}\mathcal{C}$ in the sense of Power and Robinson [PR97, Pow02], and a map F makes the square commute for all J iff F is central (Definition 3.4(b)). Thielecke, Führmann and Selinger begin their development from $\mathsf{H}\mathcal{C}$ as a premonoidal category, whereas we have constructed it as an intermediate stage on the journey from $\mathcal{C}$ to $\mathsf{S}\mathcal{C}$.

6.9. Definition. $\widehat{F} : X \longrightarrow\!\!\!\!\!\times\; Y$ is ***discardable*** or ***copyable*** respectively if it respects the naturality of the terminal (or product) projection (!) and the diagonal (Δ).

These terms are due to Hayo Thielecke [Thi97a, Definition 4.2.4], who demonstrated their computational meaning (*op. cit.*, Chapter 6). In particular, non-terminating programs are not discardable, but he gave examples of programs involving control operators that are discardable but not copyable, so both of these properties are needed for a program to be free of control effects such as jumps (Remark 1.5). In fact, these conditions are enough to characterise first class maps in the topological interpretation [F], but not for general computational effects [Füh02].

6.10. Lemma. *Δ is natural in $\mathsf{S}\mathcal{C}$, i.e. all first class maps are copyable.*

$$\begin{tikzcd}
X \times X \arrow[r, "\widehat{H} \times X"] & Y \times X \arrow[r, "Y \times \widehat{H}"] & Y \times Y \\
X \arrow[u, "\Delta_X"] \arrow[rr, "\widehat{H}"] \arrow[urr, phantom, "?"] & & Y \arrow[u, "\Delta_Y"']
\end{tikzcd}$$

Proof. As in Lemma 3.7, we show that the above diagram commutes by making it into a cube together with

$$\begin{tikzcd}
\Sigma^2 X \times \Sigma^2 X \arrow[r, "\Sigma^H \times \Sigma^2 X"] & \Sigma^2 Y \times \Sigma^2 X \arrow[r, "\Sigma^2 Y \times \Sigma^H"] & \Sigma^2 Y \times \Sigma^2 Y \\
\Sigma^2 X \arrow[u, "\Delta_{\Sigma^2 X}"] \arrow[rr, "\Sigma^H"] & & \Sigma^2 Y \arrow[u, "\Delta_{\Sigma^2 Y}"']
\end{tikzcd}$$

which commutes by naturality of Δ in $\mathcal{C}$, as do the side faces of the cube, the other edges being η_X, $\eta_X \times \eta_Y$, *etc.* The top and bottom faces commute because H is a homomorphism and by naturality of $H^{(-)}$ and $(\Sigma^2 H)^{(-)}$ with respect to η_X and η_Y. The original diagram therefore commutes because $\eta_Y \times \eta_Y$ is mono by Corollary 6.4. ∎

6.11. Proposition. *$\mathsf{S}\mathcal{C}$ has finite products and $\mathcal{C} \to \mathsf{S}\mathcal{C}$ preserves them.*

Proof. The terminal object $\mathbf{1}$ is preserved since Σ is the initial algebra (Lemma 3.12). The product projections and diagonals are inherited from $\mathcal{C}$, so $\Delta ; p_0 = \mathsf{id} = \Delta ; p_1$ and $\Delta ; (p_0 \times p_1) = \mathsf{id}$.

Then, for $\mathsf{S}\mathcal{C}$-maps $a = \widehat{H} : \Gamma \to X$ and $b = \widehat{K} : \Gamma \to Y$, we obtain $\langle a, b\rangle$ as $\Delta \,;\, a \times b$, but we need centrality (Lemma 3.7) to make $a \times b$ well defined.

$$\begin{tikzcd}
 & \Gamma \arrow[r, "a"] \arrow[dr, phantom, "?"] & X \\
\Gamma \arrow[ur] \arrow[r, "\Delta"] \arrow[dr] & \Gamma \times \Gamma \arrow[u, "p_0"'] \arrow[d, "p_1"] \arrow[r, "a \times b"] \arrow[dr, phantom, "?"] & X \times Y \arrow[u, "p_0"'] \arrow[d, "p_1"] \\
 & \Gamma \arrow[r, "b"] & Y
\end{tikzcd}$$

The issue is that the squares commute. In $\mathcal{C}$, the upper one is

$$\begin{tikzcd}
\Sigma^\Gamma \arrow[r, equal, "\mathsf{id}^\Gamma"] \arrow[d, "(\Sigma^{!_\Gamma})^\Gamma \equiv \Sigma^{p_0}"'] \arrow[dr, phantom, "?"] & \Sigma^\Gamma \arrow[d, "(\Sigma^{!_Y})^\Gamma"] & \Sigma^X \arrow[l, "H"'] \arrow[d, "(\Sigma^{!_Y})^X"] \\
\Sigma^{\Gamma \times \Gamma} & \Sigma^{\Gamma \times Y} \arrow[l, "K^\Gamma"'] & \Sigma^{X \times Y} \arrow[l, "H^Y"']
\end{tikzcd}$$

using one of the two definitions for $\widehat{H} \times \widehat{K} = a \times b$. The left-hand square commutes by Lemma 3.7 because $K : \Sigma^Y \to \Sigma^\Gamma$ is discardable (as it is a homomorphism), and the right-hand square by naturality of $H^{(-)} : \Sigma^{X \times (-)} \to \Sigma^{\Gamma \times (-)}$ with respect to $! : Y \to \mathbf{1}$.

For uniqueness, suppose that $f \,;\, p_0 = a$ and $f \,;\, p_1 = b$. Then

$$\begin{array}{rcll}
f & = & f \,;\, \Delta_{X \times Y} \,;\, (p_0 \times p_1) & \\
 & = & \Delta_\Gamma \,;\, (f \times f) \,;\, (p_0 \times p_1) & \text{naturality of } \Delta \\
 & = & \Delta_\Gamma \,;\, \big((f \,;\, p_0) \times (f \,;\, p_1)\big) & \\
 & = & \Delta_\Gamma \,;\, (a \times b) & \times \text{ is a functor on } \mathsf{S}\mathcal{C} \\
 & = & \langle a, b\rangle & \blacksquare
\end{array}$$

7. The structure of $\mathsf{S}\mathcal{C}$

We still have to show that $\mathsf{S}\mathcal{C}$ has powers of Σ, and that all of its objects are sober. In fact $\mathsf{S}\mathcal{C}$ freely adjoins sobriety to $\mathcal{C}$.

7.1. LEMMA. *Still writing $\Sigma^{(-)}$ for the exponential in $\mathcal{C}$, $\mathsf{S}\mathcal{C}(X,\Sigma^Y)\cong\mathcal{C}(X,\Sigma^Y)$, where the homomorphism $H:\Sigma^2Y\to\Sigma^X$ corresponds to the map $f:X\to\Sigma^Y$ by*

$$H=\Sigma^f \qquad \textit{and} \qquad f=\eta_X\,;\Sigma^H\,;\Sigma^{\eta_Y}.$$

PROOF. $\widehat{H}\in\mathsf{S}\mathcal{C}(X,\Sigma^Y)$ is by definition a homomorphism $H:\Sigma^2Y\to\Sigma^X$, whose double exponential transpose $P:X\to\Sigma^3Y$ has equal composites with $\Sigma^3Y\rightrightarrows\Sigma^5Y$ by Lemma 4.3, and so factors as $f:X\to\Sigma^Y$ through the equaliser by Proposition 4.9. More explicitly,

$$\eta_X\,;\Sigma^{\Sigma^f}\,;\Sigma^{\eta_Y} \;=\; f\,;\eta_{\Sigma^Y}\,;\Sigma^{\eta_Y} \;=\; f$$

by Lemma 2.10, and

$$\Sigma^f \;=\; \Sigma^{\Sigma^{\eta_Y}}\,;\Sigma^{\Sigma^H}\,;\Sigma^{\eta_X} \;=\; \Sigma^{\Sigma^{\eta_Y}}\,;\Sigma^{\eta_{\Sigma^Y}}\,;H \;=\; H$$

since H is a homomorphism. ∎

Proposition 6.11 (and the way in which objects of $\mathsf{S}\mathcal{C}$ are named) allow us to use the product notation ambiguously in both categories. Relying on that, we can now also justify writing Σ^X for powers in either $\mathcal{C}$ or $\mathsf{S}\mathcal{C}$.

7.2. COROLLARY. *$\mathsf{S}\mathcal{C}$ has powers of Σ and $\mathcal{C}\to\mathsf{S}\mathcal{C}$ preserves them.*

PROOF. $\mathsf{S}\mathcal{C}(\Gamma\times_{\mathsf{S}\mathcal{C}}X,\Sigma)=\mathsf{S}\mathcal{C}(\Gamma\times_{\mathcal{C}}X,\Sigma)$ by Proposition 6.11. Then by the Lemma this is $\mathcal{C}(\Gamma\times X,\Sigma)\cong\mathcal{C}(\Gamma,\Sigma^X)\cong\mathsf{S}\mathcal{C}(\Gamma,\Sigma^X)$. ∎

7.3. LEMMA. *$\mathcal{A}_{\mathsf{S}\mathcal{C}}\cong\mathcal{A}_{\mathcal{C}}$.*

PROOF. $\mathcal{A}_{\mathsf{S}\mathcal{C}}$ is defined from $\mathsf{S}\mathcal{C}$ in the same way as $\mathcal{A}_{\mathcal{C}}\equiv\mathcal{A}$ is defined from $\mathcal{C}$ (Definition 3.2). Consider the defining square for a homomorphism over $\mathsf{S}\mathcal{C}$:

$$\begin{tikzcd} A & \Sigma^2A \arrow[l, "\alpha"'] \\ B \arrow[u, "H"] & \Sigma^2B \arrow[l, "\beta"] \arrow[u, "\Sigma^2H"'] \end{tikzcd}$$

The vertices are retracts of powers of Σ, and Lemma 7.1 extends to such objects. Hence the $\mathsf{S}\mathcal{C}$-maps α, β and H might as well just be $\mathcal{C}$-maps, by Lemma 7.1, and the equations hold in $\mathsf{S}\mathcal{C}$ iff they hold in $\mathcal{C}$. ∎

7.4. Proposition. *All objects of* $\mathsf{S}\mathcal{C}$ *are sober,* $\mathsf{SS}\mathcal{C} \cong \mathsf{S}\mathcal{C}$ *and* $\mathsf{HS}\mathcal{C} \cong \mathsf{H}\mathcal{C}$*.*

Proof. The categories all share the same objects, and by Lemma 7.1,

$$\mathsf{HS}\mathcal{C}(X,Y) \;=\; \mathsf{S}\mathcal{C}(\Sigma^Y,\Sigma^X) \;\cong\; \mathcal{C}(\Sigma^Y,\Sigma^X) \;=\; \mathsf{H}\mathcal{C}(X,Y).$$

Lemma 7.3 provides the analogous result for $\mathsf{S}\mathcal{C}$, namely

$$\mathsf{SS}\mathcal{C}(X,Y) \;=\; \mathcal{A}_{\mathsf{S}\mathcal{C}}(\Sigma^Y,\Sigma^X) \;\cong\; \mathcal{A}(\Sigma^Y,\Sigma^X) \;=\; \mathsf{S}\mathcal{C}(X,Y).$$

Then all objects of $\mathsf{S}\mathcal{C}$ are sober by Theorem 4.10. ∎

By Corollary 4.12, $\mathsf{S}\mathcal{C}$ therefore has $\mathsf{focus}\,P$ for every prime P. The construction in the previous section shows that this is given by composition with the second class map force.

7.5. Lemma.
(a) *Each* $\widehat{H} : X \longrightarrow\!\!\!\!\times\; Y$ *in* $\mathsf{H}\mathcal{C}$ *is* $P\,;\mathsf{force}_Y$ *for some unique* $P : X \to \Sigma^{\Sigma^Y}$ *in* $\mathcal{C}$*.*
(b) $\widehat{H} : X \to Y$ *is in* $\mathsf{S}\mathcal{C}$ *iff* P *is prime.*
(c) *In this case,* $\mathsf{focus}\,P = P\,;\mathsf{force}_Y$*.*
(d) *On the other hand,* $x : X \vdash P = \eta_X(x) : \Sigma^{\Sigma^X}$ *is always prime, and* $\mathsf{focus}(\eta_X x) = x$*.*

$$\begin{tikzcd}
 & \Sigma^2 Y \arrow[d, shift left, "\mathsf{force}_Y"] \\
X \arrow[ur, "P"] \arrow[r, dotted, "\widehat{H} \equiv \mathsf{focus}\,P"'] & Y \arrow[u, shift left, "\eta_Y"]
\end{tikzcd}$$

Proof. The correspondence between H and P is double exponential transposition (Proposition 2.11), and H is a homomorphism (*i.e.* $\widehat{H}$ is in $\mathsf{S}\mathcal{C}$) iff P is prime, by Lemma 4.3. In particular, $H = \mathsf{id}_{\Sigma^X}$ corresponds to $P = \eta_X$. When H is a homomorphism we have

$$P \;=\; \eta_X\,;\Sigma^H \;=\; \eta_X\,;\Sigma^2\widehat{H} \;=\; \widehat{H}\,;\eta_Y,$$

or $\mathsf{thunk}(a)$, where $x : X \vdash a : Y$ is the term corresponding to $\widehat{H}$, so

$$\mathsf{focus}\,P \;=\; \mathsf{focus}(\mathsf{thunk}\,a) \;=\; a.$$ ∎

7.6. Theorem. $\mathsf{S}\mathcal{C}$ *is, up to isomorphism, the universal way of forcing all objects of* $\mathcal{C}$ *to be sober.*

$$\begin{tikzcd}
\mathsf{S}\mathcal{C} \arrow[r, dotted] & \mathcal{D} \\
\mathcal{C} \arrow[u] \arrow[ur, "F"] &
\end{tikzcd}$$

Proof. Let $\mathcal{D}$ be a category with products and an exponentiating object $\Sigma_{\mathcal{D}}$, and let $F : \mathcal{C} \to \mathcal{D}$ be a functor that preserves this structure. Suppose that all objects of $\mathcal{D}$ are sober. Given any homomorphism $H : \Sigma^Y \to \Sigma^X$ in $\mathcal{C}$, consider its image under the functor in $\mathcal{D}$. This is a homomorphism $FH : \Sigma_{\mathcal{D}}^{FY} \to \Sigma_{\mathcal{D}}^{FX}$ since F commutes with $\Sigma^{(-)}$ and preserves the Eilenberg–Moore equation. Therefore FH is of the form $\Sigma_{\mathcal{D}}^{g}$ for some unique $g : FX \to FY$, since all objects of $\mathcal{D}$ are sober. Then g is the effect of F on the given $\mathsf{S}\mathcal{C}$-morphism $\widehat{H} : X \to Y$.

This construction preserves identities and compositions by the usual uniqueness arguments, and similarly if $H = \Sigma^f$ with $f : X \to Y$ in $\mathcal{C}$ then $g = Ff$. Hence we have a commutative triangle of functors. As the objects X and Y of $\mathsf{S}\mathcal{C}$ are just objects of $\mathcal{C}$ and $F(X \times Y) \cong FX \times FY$ on $\mathcal{C}$, products in $\mathsf{S}\mathcal{C}$ are also preserved, as are powers of Σ. ■

7.7. Remark. Peter Selinger, for whom (his version of) the computational category $\mathsf{H}\mathcal{C}$ is of primary interest, calls $\mathcal{C}$ and $\mathsf{S}\mathcal{C}$ *value categories*, and takes an egalitarian view of them [Sel01, Section 3.5]. However, we have just shown that $\mathsf{S}\mathcal{C}$ has a *universal property*, so it is the ***sober completion*** of $\mathcal{C}$, and such (established) language does make a *value-judgement*: we regard $\mathsf{S}\mathcal{C}$ as *better* than $\mathcal{C}$, since it includes denotational values that we have argued *ought* to be present.

Be careful, however, to distinguish this sober completion of the *category* $\mathcal{C}$ from the sobrification $\overline{X}$ of the *space* (object) X [Joh82, Corollary II 1.7(ii)]. If $\mathcal{C}$ has equalisers, $\overline{X}$ is obtained by *forming* the equaliser that we used to *define* sobriety (*cf.* Remark 4.8). In [B] we shall obtain the space $\mathsf{pts}(A, \alpha)$ of points of an arbitrary algebra by forming an equaliser of this kind. These “concrete” constructions on objects are carried out within a *single* sufficiently expressive category, whereas $\mathsf{S}\mathcal{C}$ is a new category that is obtained “abstractly” by re-naming features of the old category $\mathcal{C}$.

7.8. Remark. What of the extra structure in Remarks 2.4ff? The lattice operations $\top$, $\bot$, $\wedge$ and $\vee$, being morphisms $\mathbf{1} \to \Sigma$ or $\Sigma \times \Sigma \to \Sigma$ in $\mathcal{C}$, are carried by the functor $\mathcal{C} \to \mathsf{S}\mathcal{C}$ into the new category. The equations for a distributive lattice still hold, because any functor preserves equations, and this one also preserves products. The Euclidean principle remains valid in the new category too, as Σ^Σ is also preserved. This leaves $\mathbb{N}$, from which preservation of the existential quantifier and continuity axiom follow easily.

The only issue is in fact the way in which *new* values are created in $\mathsf{S}\mathcal{C}$ by the *combination* of focus and primitive recursion. We leave the reader to add parameters: $\widehat{S} : \Gamma \times \mathbb{N} \times X \to X$.

7.9. Proposition. *$\mathcal{C} \to \mathsf{S}\mathcal{C}$ preserves the natural numbers object, i.e. $\mathbb{N}$ admits primitive recursion in $\mathsf{S}\mathcal{C}$.*

$$\begin{tikzcd}
& \Gamma \times \mathbb{N} \arrow[dd, dotted, "r"] & \Gamma \times \mathbb{N} \arrow[l, "+1"'] \arrow[dd, dotted, "r"] \\
\Gamma \arrow[ur, "0"] \arrow[dr, "z"'] & & \\
& X & X \arrow[l, "\widehat{S}"']
\end{tikzcd}$$

Proof. The recursion data consist of $z : \Gamma \to X$ and a homomorphism $S : \Sigma^X \to \Sigma^X$. So $Z \equiv z ; \eta_X$ is prime and has equal composites in the lower triangle below, whilst the parallel squares each commute, by naturality of η.

$$\begin{tikzcd}
& \Gamma \times \mathbb{N} \arrow[d, dotted, "R"] & \Gamma \times \mathbb{N} \arrow[l, "+1"'] \arrow[d, dotted, "R"] \\
\Gamma \arrow[ur, "0"] \arrow[r, "Z"] \arrow[dr] & \Sigma^{\Sigma^X} \arrow[d, shift right, "\eta_{\Sigma^2 X}"'] \arrow[d, shift left, "\Sigma^2 \eta_X"] & \Sigma^{\Sigma^X} \arrow[l, "\Sigma^S"'] \arrow[d, shift right, "\eta_{\Sigma^2 X}"'] \arrow[d, shift left, "\Sigma^2 \eta_X"] \\
& \Sigma^4 X & \Sigma^4 X \arrow[l, "\Sigma^3 S"']
\end{tikzcd}$$

As $\mathbb{N}$ has the universal property in $\mathcal{C}$, there are mediators $R : \Gamma \times \mathbb{N} \to \Sigma^2 X$ and $\Gamma \times \mathbb{N} \to \Sigma^4 X$ making the whole diagram commute. But, by uniqueness of the second, the composites $\Gamma \times \mathbb{N} \rightrightarrows \Sigma^4 X$ are equal, so R is prime by Lemma 4.3, and $\mathsf{focus}\, R : \mathbb{N} \to X$ is the required mediator in $\mathsf{S}\mathcal{C}$. ∎

Finally, we note a result that would hold automatically if $\mathsf{S}\mathcal{C}$ were a cartesian closed category.

7.10. Lemma. *The functor $\mathsf{S}\mathcal{C} \to \mathcal{C}^{\mathsf{op}}$ preserves such colimits as exist.*

Proof. The diagram for a colimit in $\mathsf{S}\mathcal{C}$ is a diagram for a limit in $\mathcal{C}$ whose vertices are powers of Σ and whose edges are homomorphisms. If the diagram has a colimit C in $\mathsf{S}\mathcal{C}$ then it is a cone of homomorphisms in $\mathcal{C}$ with vertex Σ^C whose limiting property is tested by other cones of homomorphisms *from powers of* Σ. We have to extend this property to cones from arbitrary objects Γ of $\mathcal{C}$.

$$\begin{tikzcd}
\Sigma^2\Gamma \arrow[r, "\Sigma^2\phi"] & \Sigma^3 Y \arrow[r, "\Sigma^2 H"] \arrow[d, "\Sigma^{\eta_Y}"'] & \Sigma^3 Z \arrow[d, "\Sigma^{\eta_Z}"] \\
\Gamma \arrow[u, "\eta_\Gamma"] \arrow[r, "\phi"] & \Sigma^Y \arrow[r, "H"] & \Sigma^Z
\end{tikzcd}$$

Let $\phi : \Gamma \to \Sigma^Y$ be a typical edge of the cone and $H : \Sigma^Y \to \Sigma^Z$ an edge of the diagram. Then $\Sigma^2\phi\,;\Sigma^{\eta_Y} : \Sigma^2\Gamma \to \Sigma^Y$ is a homomorphism, and is an edge of a cone with vertex $\Sigma^2\Gamma$ because the diagram above commutes.

Hence there is a mediator $\Gamma \to \Sigma^2\Gamma \to \Sigma^C$ to the limit. It is unique because any other such mediator $\Gamma \to \Sigma^C$ can be lifted to a homomorphism $\Sigma^2\Gamma \to \Sigma^C$ in the same way, and this must agree with the mediator that we have. ∎

7.11. PROPOSITION. *The functor $X \times (-)$ preserves (distributes over) such colimits as exist in* $\mathsf{S}\mathcal{C}$.

PROOF. The functor $X \times (-)$ on $\mathsf{S}\mathcal{C}$ is $(-)^X$ on $\mathcal{C}$, which is defined at powers of Σ and homomorphisms between them. If C is the colimit of a diagram with typical edge $\widehat{H} : Z \to Y$ in $\mathsf{S}\mathcal{C}$ then the Lemma says that Σ^C is the limit of the diagram with typical edge $H : \Sigma^Y \to \Sigma^Z$ in $\mathcal{C}$. Now $(-)^X$, in so far as it is defined, preserves limits, since it has a left adjoint $X \times (-)$ in $\mathcal{C}$. (You may like to draw the diagrams to show this explicitly.) Hence $\Sigma^{C\times X}$ is the limit of the diagram with typical edge $H^X : \Sigma^{X\times Y} \to \Sigma^{X\times Z}$ in $\mathcal{C}$. Since fewer (co)cones have to be tested, $C \times X$ is the colimit of the diagram with typical edge $X \times \widehat{H} : X \times Y \to X \times Z$ in $\mathsf{S}\mathcal{C}$. ∎

8. A lambda calculus for sobriety

In this section we show that $\mathsf{S}\mathcal{C}$ interprets the restricted λ-calculus, together with the new operation focus. For reference, we first repeat the equation in Remark 4.11.

8.1. DEFINITION. $\Gamma \vdash P : \Sigma^{\Sigma^X}$ is ***prime*** if $\Gamma,\ \mathcal{F} : \Sigma^3 X \ \vdash\ \mathcal{F}P \ =\ P\big(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)\big)$.

8.2. DEFINITION. The ***sober λ-calculus*** is the restricted λ-calculus (Definition 2.1) together with the additional rules

$$\frac{\Gamma \vdash P : \Sigma^{\Sigma^X} \qquad P \text{ is prime}}{\Gamma \ \vdash\ \mathsf{focus}\, P : X} \qquad \mathsf{focus}\, I$$

$$\frac{\Gamma \vdash P : \Sigma^{\Sigma^X} \qquad P \text{ is prime}}{\Gamma,\ \phi : \Sigma^X \ \vdash\ \phi(\mathsf{focus}\, P) \ =\ P\phi : \Sigma} \qquad \mathsf{focus}\, \beta$$

$$\frac{\Gamma \vdash a, b : X \qquad \Gamma,\ \phi : \Sigma^X\ \vdash\ \phi a = \phi b}{\Gamma\ \vdash\ a = b}\ \mathrm{T}_0$$

The definition $\mathsf{thunk}\, a = \eta_X(a) = \lambda\phi.\, \phi a$ serves as the elimination rule for focus. Using this, equivalent ways of writing the $\mathsf{focus}\,\beta$ and η (T_0) rules are

$$\mathsf{thunk}\,(\mathsf{focus}\, P) = P \qquad \text{and} \qquad \mathsf{focus}\,(\mathsf{thunk}\, x) = x,$$

where P is prime.

8.3. Remark. The restriction of focus to primes is the crucial difference from Thielecke's force calculus, and is the reason why we gave it a new name. In the $\mathsf{focus}\,\beta$-rule, how can we tell how much of the *surrounding* expression is the predicate ϕ that is to become the *sub-term* of P? For example, for $F : \Sigma^\Sigma$,

$$\text{does}\quad F\big(\phi(\mathsf{focus}\, P)\big) \quad \text{reduce to} \quad F(P\phi) \quad \text{or to} \quad P(\phi\,;F)?$$

So long as P is prime, it doesn't matter, because these terms are equal. In Remark 11.4 we consider briefly what happens if this side-condition is violated.

Proof. The double transpose $H : \Sigma^X \to \Sigma^\Gamma$ of P is a homomorphism with respect to the double transpose $J : \Sigma \to \Sigma^\Gamma$ of F. ∎

The interaction of focus with *substitution*, *i.e. cut elimination*, brings no surprises.

8.4. Lemma. *If $\Gamma \vdash P : \Sigma^2 X$ is prime then so is $\Delta \vdash (u^* P) : \Sigma^2 X$ for any substitution $u : \Delta \to \Gamma$ [Tay99, Section 4.3], and then*

$$\Delta\ \vdash\ u^*(\mathsf{focus}\, P)\ =\ \mathsf{focus}(u^* P) : X.$$

Proof. In the context $[\Delta,\ \mathcal{F} : \Sigma^3 X]$, since x, ϕ and $\mathcal{F}$ don't depend on Γ,

$$\mathcal{F}(u^* P)\ \equiv\ u^*(\mathcal{F} P)\ =\ u^*\big(P(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x))\big)\ \equiv\ (u^* P)\big(\lambda x.\, \mathcal{F}(\lambda\phi.\, \phi x)\big),$$

so $u^* P$ is prime. Then, in the context $[\Delta,\ \phi : \Sigma^X]$,

$$\phi\big(u^*(\mathsf{focus}\, P)\big)\ \equiv\ u^*\big(\phi(\mathsf{focus}\, P)\big)\ =\ u^*(P\phi)\ \equiv\ (u^* P)\phi\ =\ \phi\big(\,\mathsf{focus}(u^* P)\big)$$

using the $\mathsf{focus}\,\beta$-rule, whence the substitution equation follows by T_0. ∎

8.5. Theorem. *$\mathsf{S}\mathcal{C}$ is a model of the sober λ-calculus.*

Proof. Since $\mathsf{S}\mathcal{C}$ has products and powers of Σ (Proposition 6.11 and Corollary 7.2), it is a model of the restricted λ-calculus. Lemma 7.5 provides the interpretation of $\mathsf{focus}\, P$ and (the second form of) its β- and η-rules. ∎

8.6. Remark. Let $\mathcal{C}$ and $\mathcal{D}$ be the categories corresponding to the restricted and sober λ-calculi respectively, as in Remark 2.7. Since the calculi have the same types, and so contexts, $\mathcal{D}$ has the same objects as $\mathcal{C}$ and $\mathsf{S}\mathcal{C}$.

We have shown how to interpret focus in $\mathsf{S}\mathcal{C}$, and that the equations are valid there. Hence we have the interpretation functor $[\![-]\!] : \mathcal{D} \to \mathsf{S}\mathcal{C}$ in the same way as in Proposition 2.8, where $[\![-]\!]$ acts as the identity on objects (contexts).

Since $\mathcal{D}$ interprets focus by definition, all of the objects of $\mathcal{D}$ are sober by Corollary 4.12. The universal property of $\mathsf{S}\mathcal{C}$ (Theorem 7.6) then provides the inverse of the functor $[\![-]\!]$, making it an isomorphism of categories.

Alternatively, we can show directly that $\mathcal{D}$ has the same morphisms as $\mathsf{S}\mathcal{C}$, by proving a normalisation result for the sober λ-calculus. Besides being more familiar, this approach demonstrates that we have stated all of the necessary equations in the new calculus. We have already shown that the new calculus is a *sound notation* for morphisms of the category $\mathsf{S}\mathcal{C}$, and it remains to show that this notation is *complete*.

We can easily extract any sub-term $\mathsf{focus}\, P$ from a term of power type:

8.7. Lemma. $\phi[\mathsf{focus}\, P] = P(\lambda x.\, \phi[x])$.

Proof. $[\mathsf{focus}\, P/x]^*\phi[x] = (\lambda x.\, \phi[x])(\mathsf{focus}\, P)$, where $[\ ]^*$ denotes substitution. ∎

The term ϕ in $\mathsf{focus}\,\beta$ may itself be of the form $\mathsf{focus}\, P$:

8.8. Lemma. *Let $\Gamma \vdash P : \Sigma^3 X$ (sic) and $\Gamma \vdash Q : \Sigma^2 X$ be primes. Then*

$$(\mathsf{focus}\, P)(\mathsf{focus}\, Q) = PQ \qquad \textit{and} \qquad \mathsf{focus}\, P = \lambda x.\, P(\lambda\phi.\, \phi x).$$

This equation for $\mathsf{focus}\, P$ is the one in Proposition 4.9 and Lemma 7.1.

Proof. $(\mathsf{focus}\, P)(\mathsf{focus}\, Q) = Q(\mathsf{focus}\, P) = PQ$ using $\mathsf{focus}\,\beta$ twice.

In particular, put $Q = \eta_X(x) = \lambda\phi.\, \phi x$, so $x = \mathsf{focus}\, Q$. Then

$$(\mathsf{focus}\, P)x = (\mathsf{focus}\, P)(\mathsf{focus}\, Q) = PQ = P(\lambda\phi.\, \phi x),$$

from which the result follows by the $\lambda\eta$-rule. ∎

For the extra structure, we need a symbolic analogue of Proposition 7.9, now including the parameters Γ and $m : \mathbb{N}$. Note that S here is the double transpose of the same letter there.

8.9. Lemma. *Let $\Gamma \vdash Z : \Sigma^2 X$ and $\Gamma, m : \mathbb{N}, u : X \vdash S(m,u) : \Sigma^2 X$ be prime. Put*

$$\begin{aligned} \Gamma, n : \mathbb{N} \vdash r_n &= \mathsf{rec}\big(n, \mathsf{focus}\, Z, \lambda mu.\ \mathsf{focus}\, S(m,u)\big) : \Sigma^2 X \\ \Gamma, n : \mathbb{N} \vdash R_n &= \mathsf{rec}\big(n, Z, \lambda mF\phi.\, F(\lambda u.\, S(m,u)\phi)\big) : X. \end{aligned}$$

Then R_n is prime and $\Gamma, n : \mathbb{N} \vdash r_n = \mathsf{focus}\, R_n$.

PROOF. We prove

$$\Gamma,\ n:\mathbb{N}\ \vdash\ R_n\ =\ \lambda\phi.\,\phi(r_n)$$

by induction on n. For $n=0$, since Z is prime,

$$\lambda\phi.\,\phi(r_0)\ =\ \lambda\phi.\,\phi(\mathsf{focus}\,Z)\ =\ \lambda\phi.\,Z\phi = Z\ =\ R_0.$$

Suppose that $R_n = \lambda\phi.\,\phi(r_n)$ for some particular n. Then

$$\begin{array}{rcll} R_{n+1} & = & \lambda\phi.\,R_n\big(\lambda u.\,S(n,u)\phi\big) & \text{recursion step} \\ & = & \lambda\phi.\,\big(\lambda\phi'.\,\phi'(r_n)\big)\big(\lambda u.\,S(n,u)\phi\big) & \text{induction hypothesis} \\ & = & \lambda\phi.\,S(n,r_n)\phi & \lambda\beta \\ & = & \lambda\phi.\,\phi\big(\,\mathsf{focus}\,S(n,r_n)\big) & \mathsf{focus}\,\beta,\ S\ \text{prime} \\ & = & \lambda\phi.\,\phi(r_{n+1}) & \text{recursion step.} \end{array}$$

Since R_n is equal to some $\lambda\phi.\,\phi(a)$, it is prime, and the required equation follows by $\mathsf{focus}\,\eta$. ∎

8.10. PROPOSITION. *Every term $\Gamma\vdash a:X$ in the sober λ-calculus is provably equal (in that calculus) to*

(a) *some term that is already definable in the* restricted *calculus, if X is Σ^U, so a is logical in the sense of Notation 2.3; or*

(b) $\mathsf{focus}\,P$*, for some prime $\Gamma\vdash P:\Sigma^2X$ in the restricted λ-calculus, otherwise, i.e. when $X=\mathbb{N}$, so a is numerical.*

Cf. *Lemmas 7.1 and 7.5(a) respectively.*

PROOF. By structural recursion on the term a.

(a) The result is trivial for variables and constants ($\top$, $\bot$, 0), where $P=\eta_X(a)$.

(b) If $a=\lambda u.\,\phi$, $\phi\wedge\psi$, $\phi\vee\psi$ or $\exists n.\,\phi[n]$ then the recursion hypothesis says that ϕ and ψ, being logical, are provably equal to terms in the restricted calculus, whence so is a.

(c) If $a=\mathsf{focus}\,P$, the recursion hypothesis says that $P:\Sigma^2X$ is provably equal to a term in the restricted calculus (not a focus, as it is logical). Moreover, if $a:\Sigma^U$ then Lemma 8.8 rewrites it without using focus.

(d) If $a=\phi u$ then (by the recursion hypothesis, and as it is logical) ϕ is (provably equal to a term that is) defined in the restricted calculus. If $u:\Sigma^V$ then u is too. Otherwise, $u=\mathsf{focus}\,P$, and then $a=\phi(\mathsf{focus}\,P)=P\phi$ by $\mathsf{focus}\,\beta$.

(e) $\mathsf{rec}(\mathsf{focus}\,N,z,s)=\mathsf{focus}\,\big(\lambda\phi.\,N(\lambda n.\,\phi[\mathsf{rec}(n,z,s)])\big)$ by Lemma 8.7, so the first argument of rec need not involve focus.

(f) If $a=\mathsf{rec}(n,z,s):X$ then $z,s:X$, so they are provably equal to terms in the restricted calculus if $X=\Sigma^U$, whilst Lemma 8.9 rewrites a if $X=\mathbb{N}$.

(g) $\mathsf{succ}(\mathsf{focus}\,P)=\mathsf{focus}(\Sigma^2\mathsf{succ}\,P)$, where $(\Sigma^2\mathsf{succ})P$ is prime by the symbolic version of Lemma 4.6.

(h) In anticipation of Remark 9.1,

$$\big((\mathsf{focus}\,P)=_{\mathbb{N}}(\mathsf{focus}\,Q)\big)\qquad\text{may be rewritten as}\qquad P(\lambda x.\,Q(\lambda y.\,x=y)),$$

using Lemma 8.7 twice, and similarly for $\neq$. ∎

8.11. Warning. Once again, this dramatic simplification of the calculus (that focus is only needed at base types such as $\mathbb{N}$) relies heavily on the restriction on the introduction of $\mathsf{focus}\,P$ for primes only, *i.e.* on working in $\mathsf{S}\mathcal{C}$. Hayo Thielecke shows that force is needed at *all* types in the larger category $\mathsf{H}\mathcal{C}$ whose morphisms involve control operators [Thi97a, Section 6.5].

8.12. Theorem. *If $\mathcal{C}$ is the category generated by the restricted λ-calculus in Definition 2.1 then $\mathsf{S}\mathcal{C}$ is the category generated by the sober λ-calculus. If $\mathcal{C}$ has the extra structure in Remarks 2.4ff then so does $\mathsf{S}\mathcal{C}$.*

So the extension of the type theory is equivalent to the extension of the category.

$$\begin{tikzcd}
\begin{array}{c}\text{restricted}\\ \lambda\text{-calculus}\end{array} \arrow[rr, mapsto, "\text{add } \mathsf{focus}\ \&\ \beta,\eta"] \arrow[d, mapsto, "\mathsf{Cn}^{(-)}"'] & & \begin{array}{c}\text{sober}\\ \lambda\text{-calculus}\end{array} \arrow[rr, mapsto, "\text{Proposition 8.10}"] \arrow[rrd, mapsto, "\mathsf{Cn}^{(-)}"'] & & \begin{array}{c}\text{normal}\\ \text{forms}\end{array} \arrow[d, mapsto, "\mathsf{Cn}^{(-)}"] \\
\mathcal{C} \arrow[rr, mapsto] & & \mathsf{S}\mathcal{C} \arrow[rr, equal] & & \mathcal{D}
\end{tikzcd}$$

Proof. We rely on the construction of the category Cn of contexts and substitutions developed in [Tay99], and have to show that the trapezium commutes.

The categories $\mathcal{C}$, $\mathsf{S}\mathcal{C}$ and $\mathcal{D}$ in Remark 8.6 have the same objects. The morphisms of $\mathcal{C}$ and $\mathcal{D}$ are generated by weakenings and cuts, where weakenings are just product projections. A cut $[a/x] : \Gamma \to \Gamma \times X$ splits the associated product projection, and corresponds to a term $\Gamma \vdash a : X$ in the appropriate calculus, *modulo* its equations.

By Proposition 8.10, the term a of the sober calculus is uniquely of the form $\mathsf{focus}\,P$, where P is a prime defined in the restricted calculus (so the triangle commutes). Hence $[a/x]$ in $\mathcal{D}$ corresponds to the $\mathsf{S}\mathcal{C}$-morphism $\langle \mathsf{id}, \widehat{H}\rangle$, where H is the double exponential transpose of P. ∎

9. Theory of descriptions

We have seen that focus is redundant for types of the form Σ^X, since they are all sober, so $\mathbb{N}$ is the only type of the restricted λ-calculus that still needs to be considered. In fact, if it is defined to admit primitive recursion alone, as in Remark 2.5, it has *points* "missing". These may be added in various equivalent ways, using

(a) the focus operator for sobriety,

(b) definition by description in the sense of Russell,

(c) the search or minimalisation operator μ in general recursion, or

(d) the "orthogonality" mediator for repleteness.

9.1. Remark. The relevant property of $\mathbb{N}$ in the first part of the discussion is not recursion, but the fact that there are morphisms

$$\exists_{\mathbb{N}} : \Sigma^{\mathbb{N}} \to \Sigma \qquad \text{and} \qquad (=_{\mathbb{N}}) : \mathbb{N}\times\mathbb{N} \to \Sigma$$

with the expected logical properties. Objects that carry these structures are called ***overt*** and ***discrete*** respectively (Sections C 6–8).

The whole of this paper (apart from Section 5) also applies to the category of sets and functions, or to any topos. There Σ^X is the powerset, more usually written Ω^X, and $\eta_X(x)$ is the ultrafilter of subsets to which $x \in X$ belongs. All sets are overt and discrete, so the argument that follows (up to Corollary 10.5) applies to them as well as to the natural numbers. See [LS86, Section II 5] for a discussion of definition by description in a topos, the crux of which is that $\{\} : X \to \Sigma^X$ is a regular mono, *cf.* our Definition 4.7 for sobriety.

9.2. Lemma. *$[a/x]^*\phi \;=\; \exists n.\, \phi[n] \wedge (n = a)$, cf. Lemma 8.7.* ∎

9.3. Definition. A predicate $\Gamma, n : \mathbb{N} \vdash \phi[n]$ is called a ***description*** if it is uniquely satisfied, *i.e.*

$$\Gamma \;\vdash\; (\exists n.\, \phi[n]) \;=\; \top \qquad \text{and} \qquad \Gamma,\; n, m : \mathbb{N} \;\vdash\; (\phi[n] \wedge \phi[m]) \;=\; (\phi[n] \wedge n = m).$$

We shall refer to these two conditions as *existence* and *uniqueness* respectively. Using inequality, uniqueness may be expressed as

$$\Gamma \;\vdash\; (\exists m, n.\, \phi[m] \wedge \phi[n] \wedge n \neq m) \;=\; \bot.$$

9.4. Definition. Any description entitles us to *introduce* its numerical witness,

$$\frac{\Gamma, n : \mathbb{N} \;\vdash\; \phi[n] : \Sigma \qquad \text{description}}{\Gamma \;\vdash\; \mathsf{the}\, n.\, \phi[n] : \mathbb{N},}$$

the *elimination* rule being the ***singleton***

$$n : \mathbb{N} \;\vdash\; \{n\} \equiv (\lambda m.\, m = n) : \Sigma^{\mathbb{N}},$$

which is easily shown to be a description. Then the β- and η-rules are

$$\Gamma, n : \mathbb{N} \;\vdash\; (n = \mathsf{the}\, m.\, \phi[m]) \;=\; \phi[n] \qquad \text{and} \qquad n : \mathbb{N} \;\vdash\; (\mathsf{the}\, m.\, m = n) \;=\; n.$$

The restricted λ-calculus, together with the lattice structure and primitive recursion in Remarks 2.4–2.5 and these rules, is called the ***description calculus***.

9.5. Lemma. *Let $\Gamma,\; n : \mathbb{N} \;\vdash\; \psi[n] : \Sigma$ be another predicate. Then, assuming these rules,*
(a) *$\psi\big(\mathsf{the}\, n.\, \phi[n]\big) \;=\; \exists m.\, \psi[m] \wedge \phi[m]$;*
(b) *if ψ is also a description then $\big(\mathsf{the}\, n.\, \phi[n] = \mathsf{the}\, m.\, \psi[m]\big) \;=\; \exists m.\, \phi[m] \wedge \psi[m]$.*

Proof. By Lemma 9.2, $\psi\big(\mathsf{the}\, n.\, \phi[n]\big) \;=\; \exists m.\, \psi[m] \wedge \big(m \;=\; \mathsf{the}\, n.\, \phi[n]\big)$, which is $\exists m.\, \psi[m] \wedge \phi[m]$ by the β-rule for descriptions. ∎

9.6. LEMMA. $\Gamma \vdash \mathsf{rec}\big(n, z, \lambda mu.\, s(m,u)\big) = \mathsf{the}\, n.\, \rho[n]$ *where*

$$\Gamma,\ n : \mathbb{N} \vdash \rho[n] \equiv \mathsf{rec}\big(n,\ (\lambda r.\, z = r),\ \lambda m\phi r.\, \exists u.\, r = s(m,u) \wedge \phi[u]\big).$$

PROOF. In the recursion step, $\rho[r]$ is the description of the result at $n+1$ and $\phi[u]$ is the description of the sub-result $u = \mathsf{rec}(n, z, \lambda mu.\, s)$. ∎

Hence we have the analogue of Proposition 8.10 for descriptions.

9.7. PROPOSITION. *Any term $\Gamma \vdash a : X$ in the description calculus is provably equal*
(a) *to some term not involving "*the*", if it is logical, or*
(b) *to* $\mathsf{the}\, n.\, \phi[n]$ *for some description $\Gamma \vdash \phi : \Sigma^{\mathbb{N}}$, if it is numerical.*

PROOF. By structural recursion on the term a, in which Lemmas 9.2, 9.5 and 9.6 handle the non-trivial cases. ∎

9.8. REMARK. As in Remark 8.3 for focus, we must be careful about the *scope* of the description, — how much of the surrounding expression is taken as the formula ψ? For $F : \Sigma^{\Sigma}$, does

$$F\big(\psi(\mathsf{the}\, n.\, \phi[n])\big) \quad \text{reduce to} \quad F\big(\exists n.\, \phi[n] \wedge \psi[n]\big) \quad \text{or to} \quad \exists n.\, \phi[n] \wedge F(\psi[n])?$$

Once again, it does not matter, as they are equal, *so long as ϕ is a description.* Otherwise, they are different if $F(\bot) = \top$, for example if $F = \lambda\sigma.\, \top$ or (in set theory) $F = \neg$.

9.9. REMARK. The theory of descriptions was considered by Bertrand Russell [RW13, Introduction, Chapter III(1)] [vH67, pp 216–223] [GG00, Section 7.8.4]. The theme of his development is that $\mathsf{the}\, n.\, \phi[n]$ is *incomplete*: it acquires a meaning only when embedded in a predicate ψ, as in Lemma 9.5. (This came out of his dispute with Hugh McColl and Alexius Meinong regarding grammatically correct noun-phrases that don't denote [GG00, Section 7.3].) Russell defined

$$\psi\big[\mathsf{the}\, n.\, \phi[n]\big] \quad \text{as} \quad \exists n.\ \psi[n] \wedge \phi[n] \wedge (\forall m.\, \phi[m] \Rightarrow n = m),$$

incorporating the condition of unique satisfaction as a *conjunct* in this predicate. He used an inverted iota ($\rotatebox[origin=c]{180}{ι}$) for the description operator.

So long as ϕ is a description, Russell's definition is equivalent to our β-rule, but $\forall$ is not a symbol of our calculus — for the reasons that we set out in Section 1.

Gottlob Frege had treated the description operator as an everywhere-defined function-symbol, written \ [Fre93, §11] [GG00, Section 4.5.6]. He therefore had to make a case-distinction, in which $\backslash\phi$ returns the member of a singleton class, but *the class itself* if it is not a singleton. A 1960s logic textbook that I prefer not to advertise assigns $0 = \mathsf{the}\, n.\, \phi[n]$ whenever ϕ fails either of the conditions for being a description, with the result that

the unicorn *is* the author of *Principia Mathematica* is *true* since $0 = 0$ (and this book had two authors).

As we have observed, if we are allowed to write $\mathsf{the}\, n.\, \phi[n]$ without ϕ satisfying the condition, then all sorts of mathematical transformations that we would normally expect to be able to make become invalid. Frege's case-distinction is not computable — we have first to determine the cardinality of the class, which may involve answering an arbitrarily difficult mathematical question. It also illustrates the untyped nature of his calculus (which was a part of its downfall): if we introduce $a = \mathsf{the}\, x.\, \phi[x]$, we at least expect $\phi[a]$ to be *meaningful* (though maybe false if ϕ is unwitnessed), which it is not if a is a set. Even Russell's good intentions of enforcing the description property are frustrated by his *object*-language implementation, as it may result in some larger formula becoming true contrary to common sense.

Giuseppe Peano [Pea97] [GG00, Section 5.4.3] had also recognised the incomplete nature of description-phrases in mathematics. On the other hand, he required the predicate to be a description as a premise to the definition of the operator, for which he wrote $\bar{\iota}$, using ιa for our $\{a\}$. So, the condition of unique satisfaction is part of the meaning in a *meta*-logical way: if the author has written $\mathsf{the}\, n.\, \phi[n]$ anywhere, there is an implicit claim that ϕ has been proved to be a description, and this fact may be re-used anywhere in the argument. This is the point of view that we have taken: it is a side-condition on the well-formedness of the expression. At the very least, it *documents* the fact, and to rely on some exceptional behaviour is *hacking*.

Whilst Russell's extension of Peano's iota to non-descriptions is questionable, from our denotational point of view, he did take the technical analysis a step further by considering the *scope* of the expression, as in Remark 9.8.

The obvious way to find $\mathsf{the}\, n.\, \phi[n]$ is to *search* for the *least* n that satisfies $\phi[n]$. In order to be sure that it *is* the least, we must check $\neg\phi[m]$ for each $m < n$ along the way.

9.10. Definition. $\Gamma \vdash \alpha : \Sigma$ is said to be ***complemented*** or ***decidable*** if there is some (unique) $\Gamma \vdash \beta : \Sigma$ such that $\alpha \wedge \beta = \bot$ and $\alpha \vee \beta = \bot$. We write $\neg\alpha$ for β.

9.11. Lemma.
(a) *Any description ϕ on $\mathbb{N}$ is decidable, with* $\neg\phi[n] \equiv \exists m.\, \phi[m] \wedge (n \neq m)$.
(b) *Let* $\Gamma, n : \mathbb{N} \vdash \psi[n] : \Sigma$ *be a decidable predicate such that* $\Gamma \vdash \exists n.\, \psi[n] = \top$. *Then*

$$\Gamma,\ n : \mathbb{N}\ \vdash\ \phi[n]\ \equiv\ \psi[n] \wedge \forall m < n.\, \neg\psi[n]$$

is a description, and $\Gamma \vdash \mu n.\, \psi[n] \equiv \mathsf{the}\, n.\, \phi[n]$ *is the least* n *for which* $\psi[n] = \top$. ∎

9.12. Remark. The search operator μ is usually defined without the existence and decidability conditions (ψ being replaced by a partial function $\psi : \mathbb{N} \rightharpoonup \mathbf{2}$), and then itself defines a partial function, *i.e.* a program that need not terminate.

The universal property of $\mathbb{N}$, as formulated in category theory by Bill Lawvere, is known in logic as *primitive* recursion. Adding the search operation gives ***general recursion***. This is known to be properly more powerful, as functions can be defined using it that grow much faster than is possible using primitive recursion alone. Since, as we show in the next section, definition by description in $\mathbb{N}$ is equivalent to sobriety, the way that we described in Section 1 of defining computational values *via* observations really does define bigger numbers than we could obtain directly. (At any rate, it defines bigger *functions*, but since functional notation such as 10^n and its generalisations are essential for writing big numbers, it is widely argued that general recursion does indeed define bigger *numbers*.)

Although general recursive functions are partial, we would prefer to treat them as total functions $\mathbb{N} \to \mathbb{N}_\perp$ into the *lift*. This object may be seen as a closed subspace of $\Sigma^{\mathbb{N}}$, but its construction in abstract Stone duality makes rather serious use of the lattice structure on Σ [D, F].

9.13. Remark. When we add subspaces to our calculus in [B], we find that a predicate that is a description on the subspace of interest may no longer satisfy the existence and uniqueness criteria on the ambient space. Conversely, *any* predicate becomes a description when restricted to the (possibly empty) locally closed subspace defined by the $\top$ and $\perp$ equations in Definition 9.3. Nevertheless, it turns out that $\mathsf{the}\ n.\ \phi[n]$ may be manipulated on the subspace by using the β-rule as we have given it, but on the ambient space, even though this it is not a well formed expression there. Although the reduction may result in expressions with different meanings on the ambient space, they agree as intended on the subspace.

10. Sobriety and description

In this section we prove that focus and description are inter-definable, for the natural numbers. In the following notation, we need to show that ϕ is a description iff P is prime.

10.1. Lemma. *This is a retraction,*

$$\begin{tikzcd} \Sigma^{\mathbb{N}} \arrow[r, shift left, rightarrowtail, "\phi \mapsto \lambda\psi.\, \exists n.\, \phi[n] \wedge \psi[n]"] & \Sigma^{\Sigma^{\mathbb{N}}} \arrow[l, shift left, twoheadrightarrow, "\lambda n.\, P(\lambda m.\, m = n) \hookleftarrow P\, :\, \Sigma^{\{\}}"] \end{tikzcd}$$

$$\mathbb{N} \xrightarrow{\quad n \mapsto \{n\} \equiv \lambda m.\, m = n \quad} \Sigma^{\mathbb{N}}$$

Compare the connection between $\{\}$ and η in [BW85, Lemma 5.1.3] and Lemma C 6.12. ■

First, we characterise the image of $\Sigma^{\mathbb{N}} \rightarrowtail \Sigma^{\Sigma^{\mathbb{N}}}$.

10.2. Lemma. $\Gamma \vdash P : \Sigma^2\mathbb{N}$ *is of the form* $\lambda\psi.\, \exists n.\, \phi[n] \wedge \psi[n]$ *for some* $\Gamma \vdash \phi : \Sigma^{\mathbb{N}}$ *iff it preserves* $\exists_{\mathbb{N}}$. *In this case,* $\phi = \lambda n.\, P(\lambda m.\, n = m)$.

PROOF. Suppose that P preserves $\exists$. Then

$$\begin{aligned} P\psi &= P(\lambda m.\, \exists n.\, n = m \wedge \psi[n]) \\ &= \exists n.\, P(\lambda m.\, n = m \wedge \psi[n]) \\ &= \exists n.\, P(\lambda m.\, n = m) \wedge \psi[n] \end{aligned}$$

by the Euclidean principle (Remark 2.4). The other way is easy. ■

For the rest of this section, P and ϕ will be related in this way.

10.3. LEMMA. *P preserves $\top$ iff ϕ satisfies the existence condition, and P preserves $\wedge$ iff ϕ satisfies the uniqueness condition.*

PROOF. For the first, observe that $P\top = \exists n.\, \phi[n]$. For the second, if $\phi[n_1] \wedge \psi_1[n_1]$ and $\phi[n_2] \wedge \psi_2[n_2]$ then $n_1 = n_2 = n$ by uniqueness. ■

10.4. PROPOSITION. *If P is prime then ϕ is a description, and $\mathsf{the}\, n.\, \phi[n]$ satisfies the rules for $\mathsf{focus}\, P$ (Definition 8.2).*

PROOF. As P is prime, it preserves $\top$, $\wedge$ and $\exists$ because its double exponential transpose is a homomorphism, in particular with respect to $\top$, $\wedge$ and $\exists$, as in Proposition 5.5. Also, $P = \lambda\psi.\, \exists n.\, \phi[n] \wedge \psi[n]$ by the Lemma, so $P\psi = \psi\big[\mathsf{the}\, n.\, \phi[n]\big]$, which means that the β-rules agree. For the η-rules,

$$\phi = \{n\} \equiv (\lambda m.\, m = n) \qquad \text{iff} \qquad P = \mathsf{thunk}\, n \equiv \lambda\psi.\, \psi[n],$$

which are respectively a description and prime, and then $(\mathsf{the}\, m.\, \phi[m]) = (\mathsf{focus}\, P) = n$. ■

10.5. COROLLARY. *Any overt discrete object that admits definition by description is sober. In particular, all sets and all objects of any topos are sober.* ■

The converse is the case $X = \mathbb{N}$ of Theorem 5.7, that if $H : \Sigma^{\mathbb{N}} \to \Sigma^{U}$ is a frame homomorphism (*i.e.* it preserves $\top$, $\wedge$ and $\exists$) then it is an Eilenberg–Moore homomorphism. Of course, we showed that for **LKLoc**, by re-interpreting results from the literature, not for our abstract calculus. The proof below depends on (primitive) recursion, so only applies to $\mathbb{N}$ rather than to discrete overt spaces in general, although when the whole theory is in place the result will hold for them too.

But before considering $\mathbb{N}$ we have to deal with $\mathbf{2} = \{0, 1\}$. As we did not ask for this as a base type in Section 2, $\psi : \Sigma^{\mathbf{2}}$ may be replaced by $\langle \psi_0, \psi_1 \rangle \in \Sigma \times \Sigma$, and similarly for the type of P. (Alternatively, one could formulate a result with $\psi : \Sigma^{\mathbb{N}}$ to capture the same point.) See [B, Section 11] for further discussion of disjoint unions in abstract Stone duality.

10.6. Proposition. *Let $\alpha : \Sigma$ be decidable, with $\beta = \neg\alpha$ (Definition 9.11). Then*

$$P \;\equiv\; \lambda\psi.\, (\alpha \wedge \psi[0]) \vee (\beta \wedge \psi[1])$$

is prime. This justifies ***definition by cases****:* $(\mathbf{if}\ \alpha\ \mathbf{then}\ 0\ \mathbf{else}\ 1) \equiv \mathsf{focus}\, P$.

Proof. Let $P_{\alpha\beta}$ be the obvious generalisation, so

$$\gamma \wedge P_{\alpha\beta} = P_{(\gamma\wedge\alpha)(\gamma\wedge\beta)}$$

by distributivity. The equation that we have to prove for Definition 8.1 may be written

$$(\alpha \vee \beta) \wedge \mathcal{F}P_{\alpha\beta} \;=\; (\alpha \wedge \mathcal{F}P_{\top\bot}) \vee (\beta \wedge \mathcal{F}P_{\bot\top}).$$

By the Euclidean principle (Remark 2.4) and the lattice laws, we have

$$\begin{array}{rcl} \alpha \wedge \mathcal{F}P_{\alpha\beta} & = & \alpha \wedge \mathcal{F}(\alpha \wedge P_{\alpha\beta}) \\ & = & \alpha \wedge \mathcal{F}(\alpha \wedge P_{\alpha(\alpha\wedge\beta)}) \\ & = & \alpha \wedge \mathcal{F}(\alpha \wedge P_{\alpha\bot}) \\ & = & \alpha \wedge \mathcal{F}(\alpha \wedge P_{\top\bot}) \\ & = & \alpha \wedge \mathcal{F}P_{\top\bot} \end{array}$$

■

10.7. Lemma. *For any description ϕ, we define, by primitive recursion,*

$$\phi^{\geq}[0] \equiv \top \qquad \phi^{>}[n] \equiv \phi^{\geq}[n+1] \equiv \phi^{\geq}[n] \wedge \neg\phi[n]$$

$$\phi^{<}[0] \equiv \bot \qquad \phi^{\leq}[n] \equiv \phi^{<}[n+1] \equiv \phi^{<}[n] \vee \phi[n].$$

Then $\phi^{<}[n]$ has the properties of the ordinary arithmetic order, that is, $(\mathsf{the}\ m.\, \phi[m]) < n$, and similarly for the others. ■

10.8. Proposition. *If ϕ is a description then P is prime, and $\mathsf{focus}\, P$ satisfies the rules for $\mathsf{the}\ n.\, \phi[n]$.*

Proof. The idea of the proof is to define a permutation $f : \mathbb{N} \cong \mathbb{N}$ that cycles the witness to 0 and any smaller values up by 1, leaving bigger numbers alone. The function f itself is defined by (cases and) primitive recursion, but it only has an inverse with general recursion. Let

$$f(n) \;=\; \begin{cases} 0 & \text{if } \phi[n] \\ n & \text{if } \phi^{<}[n] \\ n+1 & \text{if } \phi^{>}[n] \end{cases} \qquad \delta(n,m) \;=\; \begin{cases} & m = 0 & \wedge & \phi[n] \\ \vee & m = n & \wedge & \phi^{<}[n] \\ \vee & m = n+1 & \wedge & \phi^{>}[n], \end{cases}$$

so $\delta(n,m) \iff f(n) = m$. The operations Σ^f and I, defined by

$$\begin{array}{rcl} \Sigma^f\theta & = & \lambda n.\, \exists m.\, \delta(m,n) \wedge \theta[m] \\ I\psi & = & \lambda m.\, \exists n.\, \delta(m,n) \wedge \psi[n], \end{array}$$

are mutually inverse, because, by expanding disjunctions,

$$\begin{array}{lcl}
\exists m.\, \delta(m,n) & = & \phi[n] \vee \phi^{<}[n] \vee \phi^{\geq}[n+1] \\
\exists n.\, \delta(m,n) & = & \big(m = 0 \wedge \exists n.\, \phi[n]\big) \vee (\exists n.\, m = n+1) \\
\delta(m,n_1) \wedge \delta(m,n_2) & = & (n_1 = n_2) \vee \big(\phi^{<}[m] \wedge \phi^{>}[m]\big) \\
\delta(m_1,n) \wedge \delta(m_2,n) & = & (m_1 = m_2) \vee \big(\phi[n] \wedge \phi^{<}[n]\big) \vee \big(\phi[n] \wedge \phi^{>}[n+1]\big).
\end{array}$$

Thus Σ^f is a homomorphism, as is its inverse I by Proposition 3.5. But so too is evaluation at 0, so

$$\psi \;\mapsto\; \theta[0] \;\equiv\; \exists n.\, \delta(0,n) \;\equiv\; \exists n.\, \phi[n] \wedge \psi[n]$$

is a homomorphism, and P is prime. The β- and η-rules agree as before. ∎

10.9. Remark. Sobriety says that the functor $\Sigma^{(-)} : \mathcal{C}^{\mathsf{op}} \to \mathcal{A}$ in Definition 4.1 is full and faithful. But in this proof we only used the fact that it reflects invertibility, so it suffices to assume that $\mathbb{N}$ is replete. Then f^{-1} is the diagonal mediator that is provided by the orthogonality property [Hyl91, Tay91].

$$\begin{tikzcd}
\mathbb{N} \arrow[r, "f"] \arrow[d, "\mathsf{id}"'] & \mathbb{N} \arrow[d, "n \mapsto \lambda\psi.\, I\psi n"] \arrow[dl, dotted] \\
\mathbb{N} \arrow[r, "\eta_{\mathbb{N}}"] & \Sigma^{\Sigma^{\mathbb{N}}}
\end{tikzcd}$$

10.10. Corollary. *$H : \Sigma^{\mathbb{N}} \to \Sigma^{U}$ is an Eilenberg–Moore homomorphism iff it preserves $\top$, $\wedge$ and $\exists$.* ∎

This result can be extended from $\mathbb{N}$ to higher types on the assumption of the continuity axiom (Remark 2.6). Hence there is a version of Theorem 5.7 that links the notions of homomorphism and sobriety that we have introduced entirely abstractly using the λ-calculus with those that arise from the $\mathbb{N}$-indexed lattice structure in Remarks 2.4ff. Although the proof would only require one more section, it begins to make serious use of domain-theoretic ideas, and so properly belongs in a discussion of that subject [E, F]. Besides, surprisingly much progress can be made with the development of topology *without* the extra axiom.

11. Directions

We saw in the previous two sections that the new focus operator is equivalent to definition by description. This is more familiar, both in the sense of tradition, but also in that the requirements on its data are more idiomatic: a predicate with a unique numerical solution, rather than a term of type $\Sigma^2 X$ satisfying a strange equation. Indeed, this calculus seems to be a useful denotational basis for both mathematical and computational investigations: it plays a similar role to that of the class of total recursive functions, whilst being better both as a type theory and for computation.

As it stands, it does not meet the needs of mathematicians, who expect to be able to form subtypes by means of the axiom of comprehension and other constructions such as disjoint sums. Such subtypes, specified by a comprehension-like operation, but equipped with the subspace topology, will be added to the category in [B]. However, this is ignored by computation, *i.e.* by the reduction rules for the terms.

Topological ideas such as compact Hausdorff spaces are studied in [C, E], and the partial map classifier or lift $X_\perp$ in [D, F].

11.1. Remark. For high-level computation, on the other hand, the calculus is already a serviceable functional programming language. It has as

- *types*, $\mathbf{1}$, $\mathbb{N}$ and Σ^ℓ, where ℓ is (the product of) a list of types;
- *numerical terms*, zero, successor, recursion, description and variables; and
- *logical terms*, $\top$ and $\bot$; variables; equality and inequality of numerical terms; $\wedge$, $\vee$, $\exists$ and λ with logical sub-terms; application and recursion.

Following Peter Landin [Lan64], it is useful to "sugar" λ-application $(\lambda x.\,\phi)a$ or plain substitution $[a/x]^*t$, with syntax such as

$$\mathsf{let}\ x = a\ \mathsf{in}\ t \qquad \text{or} \qquad t\ \mathsf{where}\ x = a.$$

The Y-combinator that we derived from the continuity axiom in Remark 2.6 provides recursively defined procedures. So

$$\mathsf{letrec}\ \phi(x_1, \cdots, x_n) = F\ \mathsf{in}\ t,$$

in which ϕ may be used in F as well as in t, is encoded as

$$\mathsf{let}\ \phi \ = \ \exists n.\, \mathsf{rec}(n, \bot, \lambda m\phi x_1 \cdots x_n.\, F)\ \mathsf{in}\ t.$$

The body F may contain the "recursive call" ϕ and its recursive arguments $x_1, \cdots, x_n$; these are the variables bound by the λ within rec. The numerical variables n and m count the depth of the recursion, but their values are forgotten by $\exists n$.

The development of mathematical and topological structures contributes Floyd–Hoare reasoning and special types such as $X_\perp$ to the programming applications. However, by using mathematical intuitions to construct $\mathbb{R}$ and other objects, it will also provide methods of programming (*i.e.* algorithms) for problems where no order of evaluation is naturally apparent.

11.2. Remark. How could we then compile such a program?

As we have understood the calculi in this paper *denotationally*, let us first clarify what we mean by compilation and execution. In, for example, elementary algebra, we use the rules (such as distributivity) in whatever fashion seems best, to simplify the given complicated expression into a denotationally equivalent one that lies within a subclass of *preferred* forms. In doing this, we choose *some* amongst all valid reduction paths.

The objective may be "full" execution, in which we are satisfied with nothing less than an explicit number (assuming that the expression is of type $\mathbb{N}$), and thereby risk non-termination. Alternatively it may be to re-express the program in some simpler language, removing high-level features, but without actually doing the iterations. This translation is called *compilation*, and is required always to terminate. Sometimes compilation may use η-rules and reverse β-rules that would not be used in an *execution* strategy.

11.3. Remark. Without loss of generality, the term to be compiled is of type Σ, since terms of higher type may be applied to free variables, and a numerical term $\Gamma \vdash t : \mathbb{N}$ may be handled as $\Gamma, m : \mathbb{N} \vdash m = t : \Sigma$. For example, a numerical *function* $f : \mathbb{N} \to \mathbb{N}$ is treated in the form of its *graph*, $n, m : \mathbb{N} \vdash m = f(n) : \Sigma$.

Lemmas 9.2, 9.5 and 9.6 eliminate embedded descriptions and recursion of numerical type in favour of additional existentially quantified variables, so the numerical sub-terms that remain are ordinary expressions.

Suppose at first that the term doesn't involve disjunction or recursion: any such sub-term is replaced by a logical variable and will be handled separately.

Since we have a fragment of the simply typed λ-calculus with some constants, the term strongly normalises. This eliminates λ-abstraction and application, which is a desirable property of our compiler, as both Abstract Stone Duality and the Continuation-Passing Style introduce numerous "administrative" λ-expressions of the form $\lambda\phi.\ \phi[a]$.

Any existential quantifiers may be moved to the front by renaming the bound variables (this uses the Frobenius law to get past $\wedge$). What remains is either $\top$, $\bot$, or a conjunction of sub-terms, each of which is either

- a (free) logical variable, possibly applied to arguments, or
- an equation or inequation of two numerical sub-terms.

The entire term may be existentially quantified over some numerical variables, effectively "hiding" them.

This is a pure Prolog clause (apart from the free logical variables).

The numerical equations may be normalised by *unification*. The *occurs check* must be made, since logically $\big(\exists n.\ n = f(n)\big) \equiv \bot$ if f is any non-trivial numerical expression in which the variable n is free. The result of unification serves as a *pattern* that the free numerical variables must satisfy; the pattern is incorporated into the *head* of the clause, and many of the hidden variables are eliminated in this process. The *body* consists of the other logical conjuncts.

We restore disjunction to the language by introducing a Prolog predicate-symbol for each $\vee$ sub-term, with a clause for each branch. In order to be denotationally equivalent to the original term, these disjuncts must in general be executed in *parallel*. This is because they may fail either *finitely* (because of a clash in unification) or *infinitely* by non-termination, whereas $\vee$ is meant to be commutative. In practice, the branches are usually guarded by patterns, all but one of which fail straight away.

The term $\mathsf{rec}(n, Z, \lambda mu.\, S)$ is treated like the disjunction

$$(n = 0 \wedge Z) \vee \big(\exists m.\, n = m + 1 \wedge S(m, u)\big)$$

but with an actual link to $\mathsf{rec}(m, Z, \lambda mu.\, S)$ in place of u. The circular translation from programming language to our calculus and back therefore simply introduces a hidden variable n that counts the depth of the recursion. (There is a disjunct $\bot$ that is redundant.)

For the most part, logical variables in the main program are bound to PROLOG procedures, any free ones being (illegally) undefined procedure names. However, recursion at type $\Sigma^{\Sigma^{\mathbb{N}}}$ or higher types does involve passing logical arguments to recursive procedures. In simple cases, this may be done by Gödel-numbering them, and in fact it is not difficult to write a self-interpreter for and in pure PROLOG.

11.4. REMARK. Turning from low- to high-level programming, what can we make of Thielecke's force operator? In our treatment, we insisted that focus be accompanied by its side condition (that it only be applied to primes), so maybe we are unable to interpret control operators. But the difference is merely that we have investigated $\mathsf{S}\mathcal{C}$, which includes general recursive function, and argued that it should be used in place of $\mathcal{C}$, which only has primitive recursion. The category $\mathsf{H}\mathcal{C}$ with control operators is still there, and $\mathsf{H}\mathcal{C} \cong \mathsf{HS}\mathcal{C}$. Its terms are interpreted contravariantly in $\mathcal{C}$, by means of a λ-translation which, when written on paper, may seem complicated [Fis93, Section 4], but is dissolved away by our λ-normalising compiler.

John Reynolds has given a nice historical survey of mathematical and computational calculi that use continuations [Rey93]; for a formal introduction to control operators, you should see the work cited there. Here, we shall just say something about the consequences of dropping the primality side-condition, by way of an introduction addressed to *mathematical* readers. Note that the programming language for $\mathsf{H}\mathcal{C}$ that we're about to describe is to be *translated* into the sober calculus, and is not an *extension* of it.

In return for allowing force to be used without restriction, we have to constrain the reduction rules in general, *i.e.* to impose an order of evaluation. We choose *call by value*, in which the argument a is reduced before the β-redex $(\lambda x.\, \phi)a$, and unapplied abstractions $\lambda x.\, \phi$ are not reduced at all.

This means, in particular, that the argument of $\phi(\mathsf{force}\, P)$ gets evaluated before ϕ, turning the expression into $P\phi$. However, we must specify how much of the enclosing expression ϕ is to be consumed by this reduction. We do this by introducing another keyword, label, as a delimiter (it has no computational effect in itself). Since force may occur repeatedly, we must *name* the label that delimits each force. Assuming that neither F nor ϕ can be reduced on its own,

$$F\big(\,\mathsf{label}_k\, \phi(\mathsf{force}_k\, P)\big) \quad \text{reduces to} \quad F(P\phi).$$

What has happened here, in programming terms? The part of the continuation (ϕ) that is bracketed by label and force has been given to P as an argument. Because of the call-by-value rules, ϕ does not get executed until P puts it into an active position in front of an argument. P may also duplicate or lose ϕ. When it has finished, P does not return in the normal way to its calling context (ϕ), but to F, *i.e.* to the position of the matching label. In other words, force_k *jumps* to label_k. Unless, that is, ϕ gets executed and itself performs a different jump.

11.5. REMARK. Our compiler translated disjunction into alternative PROLOG clauses, which ought in general to be executed in parallel. If, however, one branch fails finitely, it can *back-track* to the point of choice, and proceed with another option.

Continuations provide the natural way in which to do this. Instead of having a single calling context ϕ to which it always returns normally, a sub-program that has a notion of finite failure can be supplied with *two* continuations, ϕ^+ and ϕ^-, which it may invoke respectively in the event of success and failure [Hay87, Thi01]. This translation is not the one that we obtain from *disjunction* (*cf.* Proposition 10.6), but does fall naturally out of the interpretation of *coproducts* (disjoint unions) in [B, Section 11].

11.6. REMARK. There are several reasons why pure PROLOG should arise as the intermediate or object language of our compiler, that is, that we use *logic* rather than *functional* programming. Primarily, it is that we chose Σ^X rather than $X_\perp$ our basic type constructor. Then terms $\Gamma \to \Sigma^X$ are *relations*, whereas those $\Gamma \to X_\perp$ are partial functions.

However, it is a curious feature of PROLOG that single clauses are *static*: the control and data-flow only acquire a direction when the clauses are put together into a whole program. This may perhaps be connected with the fact that it is a natural target of the continuation-passing style, *i.e.* that we are translating high-level programs that are themselves ambivalent about their direction.

References

[AGV64] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier, editors. *Séminaire de Géometrie Algébrique, IV: Théorie des Topos*, numbers 269–270 in Lecture Notes in Mathematics. Springer-Verlag, 1964. Second edition, 1972.

[App92] Andrew Appel. *Compiling with Continuations.* Cambridge University Press, 1992.

[Bou66] Nicolas Bourbaki. *Topologie Générale.* Hermann, 1966. Chapter I, "Structures Topologiques". English translation, "General Topology", distrubuted by Springer-Verlag, 1989.

[BW85] Michael Barr and Charles Wells. *Toposes, Triples, and Theories.* Springer-Verlag, 1985.

[CPR91] Aurelio Carboni, Maria-Cristina Pedicchio, and Giuseppe Rosolini, editors. *Proceedings of the 1990 Como Category Conference*, number 1488 in Lecture Notes in Mathematics. Springer-Verlag, 1991.

[Dij76] Edsger Dijkstra. *A Discipline of Programming.* Prentice–Hall, 1976.

[Eck69] Beno Eckmann, editor. *Seminar on Triples and Categorical Homology Theory*, number 80 in Lecture Notes in Mathematics. Springer-Verlag, 1969.

[Fis93] Michael Fischer. Lambda-calculus schemata. *Lisp and Symbolic Computation*, 6:259–288, 1993.

[Fox45] Ralph Fox. On topologies for function-spaces. *Bulletin of the American Mathematical Society*, 51:429–32, 1945.

[Fre93] Gottlob Frege. *Grundgesetze der Arithmetik*. 1893. English translation, *The Basic Laws of Arithmetic*, by Montgomery Furth, University of California Press, 1964.

[FS90] Peter Freyd and Andre Scedrov. *Categories, Allegories.* Number 39 in Mathematical Library. North-Holland, 1990.

[Füh99] Carsten Führmann. Direct models of the computational lambda-calculus. In *Mathematical Foundations of Programming Semantics 15*, number 20 in Electronic Notes in Theoretical Computer Science, 1999.

[Füh02] Carsten Führmann. Varieties of effects. In *Proceedings FOSSACS 2002*, number 2303 in Lecture Notes in Computer Science, pages 144–158. Springer-Verlag, 2002.

[GD71] Alexander Grothendieck and Jean Alexandre Dieudonné. *Eléments de Géometrie Algébrique, tome I: le Langage des Schémas.* Number 166 in Grundlehren der mathematische Wissenschaften. Springer-Verlag, 1971. Originally published by IHES in 1960.

[GG00] Ivor Grattan-Guinness, editor. *The Search for Mathematical Roots, 1870–1940.* Princeton University Press, 2000.

[GHK+80] Gerhard Gierz, Karl Heinrich Hoffmann, Klaus Keimel, Jimmie Lawson, Michael Mislove, and Dana Scott. *A Compendium of Continuous Lattices.* Springer-Verlag, 1980.

[Hak72] Monique Hakim. *Topos Annelés et Schémas Relatifs.* Number 64 in Ergebnisse der Mathematik und ihre Grenzgebiete. Springer-Verlag, 1972.

[Hau14] Felix Hausdorff. *Grundzüge der Mengenlehre.* 1914. Chapters 7–9 of the first edition contain the material on topology, which was removed from later editions. Reprinted by Chelsea, 1949 and 1965; there is apparently no English translation.

[Hay87] Christopher Haynes. Logic continuations. *Journal of Logic Programming*, 4:157–176, 1987.

[HM81] Karl Hofmann and Michael Mislove. Local compactness and continuous lattices. In Bernhard Banaschewski and Rudolf-Eberhard Hoffmann, editors, *Continuous Lattices*, number 871 in Lecture Notes in Mathematics, pages 209–248. Springer-Verlag, 1981.

[Hyl91] Martin Hyland. First steps in synthetic domain theory. In Carboni et al. [CPR91], pages 131–156.

[Isb75] John Isbell. Function spaces and adjoints. *Math Scand*, 36:317–39, 1975.

[Joh82] Peter Johnstone. *Stone Spaces.* Number 3 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1982.

[Kel55] John Kelley. *General Topology.* Van Nostrand, 1955. Reprinted by Springer-Verlag, Graduate Texts in Mathematics, 27, 1975.

[KP93] Max Kelly and John Power. Adjunctions whose counits are coequalisers. *Journal of Pure and Applied Algebra*, 89:163–179, 1993.

[Lan64] Peter Landin. The mechanical evaluation of expressions. *Computer Journal*, 6, 1964.

[Lin69] Fred Linton. An outline of functorial semantics. In Eckmann [Eck69], pages 7–52.

[LR73] Joachim Lambek and Basil Rattray. Localizations at injective objects in complete categories. *Proceedings of the American Mathematical Society*, 41:1–9, 1973.

[LR75] Joachim Lambek and Basil Rattray. Localizations and sheaf reflectors. *Transactions of the American Mathematical Society*, 210:279–293, 1975.

[LS86] Joachim Lambek and Philip Scott. *Introduction to Higher Order Categorical Logic.* Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician.* Number 5 in Graduate Texts in Mathematics. Springer-Verlag, 1971.

[Mog88] Eugenio Moggi. Computational lambda-calculus and monads. Technical report, LFCS, University of Edinburgh, 1988.

[Mog91] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[Pea97] Giuseppe Peano. Studii di logica matematica. *Atti Reale della Accademia degli Scienze di Torino*, 32:565–583, 1897. Reprinted in *Opere Scelte*, ed. Unione Matematica Italiana, Cremonese, Rome, 1957–9, vol. 2, pp. 201–217. English translation in *Selected Works of Giuseppe Peano* by Hubert Kennedy, Allen and Unwin, 1973, pp. 190–205.

[Plo75] Gordon Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[Plo77] Gordon Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Pow02] John Power. Premonoidal categories as categories with algebraic structure. *Theoretical Computer Science*, 278:303–321, 2002.

[PR97] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7:453–468, 1997.

[Rey93] John Reynolds. The discoveries of continuations. *Lisp and Symbolic Computation*, 6:233–247, 1993.

[RW13] Bertrand Russell and Alfred North Whitehead. *Principia Mathematica.* Cambridge University Press, 1910–13. Second edition, 1927; paperback edition to ∗56, 1962.

[Sco72] Dana Scott. Continuous lattices. In Bill Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, number 274 in Lecture Notes in Mathematics, pages 97–137. Springer-Verlag, 1972.

[Sco76] Dana Scott. Data types as lattices. *SIAM Journal on Computing*, 5:522–587, 1976.

[Sel01] Peter Selinger. Control categories and duality. *Mathematical Structures in Computer Science*, 11:207–260, 2001.

[Sim82] Harold Simmons. A couple of triples. *Topology and its Applications*, 13:201–23, 1982.

[Smy94] Michael Smyth. Topology. In Samson Abramsky et al., editors, *Handbook of Logic in Computer Science*, volume 1, pages 641–761. Oxford University Press, 1994.

[Ste78] Guy Steele. Rabbit: A compiler for Scheme. Technical Report AI TR 474, MIT, May 1978.

[Tay91] Paul Taylor. The fixed point property in synthetic domain theory. In Gilles Kahn, editor, *Logic in Computer Science 6*, pages 152–160. IEEE, 1991.

[Tay99] Paul Taylor. *Practical Foundations of Mathematics.* Number 59 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1999.

[Thi97a] Hayo Thielecke. *Categorical Structure of Continuation Passing Style.* PhD thesis, University of Edinburgh, 1997. Also available as technical report ECS-LFCS-97-376.

[Thi97b] Hayo Thielecke. Continuation semantics and self-adjointness. In *Proceedings MFPS XIII*, volume 6 of *Electronic Notes in Theoretical Computer Science.* Elsevier, 1997. URL: `http://www.elsevier.nl/locate/entcs/volume6.html`.

[Thi01] Hayo Thielecke. Comparing control constructs by double-barrelled CPS transforms. In *Seventeenth Conference on the Mathematical Foundations of Programming Semantics (MFPS17)*, Electronic Notes in Theoretical Computer Science. Elsevier Science, 2001.

[Tur35] Alan Turing. On computable numbers with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society (2)*, 42:230–265, 1935.

[vH67] Jean van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931.* Harvard University Press, 1967.

[Vic88] Steven Vickers. *Topology Via Logic.* Number 5 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988.

The papers on abstract Stone duality may be obtained from

`http://www.di.unito.it/∼pt/ASD`

[B] Paul Taylor. Subspaces in abstract Stone duality. *Theory and Applications of Categories*, 10(13):301–368, 2002.

[C] Paul Taylor. Geometric and higher order logic using abstract Stone duality. *Theory and Applications of Categories*, 7(15):284–338, 2000.

[D] Paul Taylor. Non-Artin gluing in recursion theory and lifting in abstract Stone duality, 2000.

[E] Paul Taylor. Local compactness and the Baire category theorem in abstract Sone duality, *Category Theory and Computer Science*, Ottawa, 2002.

[F] Paul Taylor. Scott domains in abstract Stone duality, 2002.

Email: `pt@di.unito.it`

This article may be accessed via WWW at `http://www.tac.mta.ca/tac/` or by anonymous ftp at `ftp://ftp.tac.mta.ca/pub/tac/html/volumes/10/12/10-12.{dvi,ps}`