Stone Duality for Monads

Richard Garner[b,1] Alyssa Renata[a,2] Nicolas Wu [a,2]

[a] *Imperial College London, London, UK*
[b] *Macquarie University, Sydney, Australia*

Abstract

We introduce a contravariant idempotent adjunction between (i) the category of ranked monads on Set; and (ii) the category of internal categories and internal retrofunctors in the category of locales. The left adjoint takes a monad T—viewed as a notion of computation, following Moggi—to its *localic behaviour category* $\mathsf{LB}T$. This behaviour category is understood as "the universal transition system" for interacting with T: its "objects" are states and the "morphisms" are transitions. On the other hand, the right adjoint takes a localic category LC—similarly understood as a transition system—to the monad $\Gamma\mathsf{LC}$ where $\Gamma\mathsf{LC}A$ is the set of A-indexed families of local sections to the source map which jointly partition the locale of objects. The fixed points of this adjunction consist of (i) *hyperaffine-unary monads*, i.e., those monads where term t admits a read-only operation $\bar{t}$ predicting the output of t; and (ii) *ample localic categories*, i.e., whose source maps are local homeomorphisms and whose locale of objects are strongly zero-dimensional. The hyperaffine-unary monads arise in earlier works by Johnstone and Garner as a syntactic characterization of those monads with Cartesian closed Eilenberg-Moore categories. This equivalence is the *Stone duality for monads*; so-called because it further restricts to the classical Stone duality by viewing a Boolean algebra B as a monad of B-partitions and the corresponding Stone space as a localic category with only identity morphisms.

Keywords: behaviour category, comodels, internal categories, internal retrofunctors, monads, stone duality

1 Introduction

Algebraic theories, as proposed by Plotkin and Power [24,25], model notions of computation. An algebraic theory consists of a set Σ of generating computational operations which define the program *syntax* along with some equations $\mathcal{E}$ that express program *semantics*. Expressing the semantics via equations means that we reason equationally about programs, but there is another way to express semantics: two programs are equivalent if they behave the same with respect to some external *environment*. Reasoning against the environment can be more intuitive than equational reasoning, but constructing an explicit model of the environment from scratch is a complicated endeavor. Faced with this tension, in this paper we explore whether there is a systematic way of recovering a model of the environment from the equations.

In logic, such explorations often manifest in the construction of a *duality*, which is a (contravariant) equivalence between some category of algebraic objects (here the algebraic theories) and some category of geometric objects (the environment). The prototypical duality is the *Stone duality* between Boolean algebras and *Stone spaces*. An element of a Boolean algebra B is typically understood as a proposition about the environment, while the elements of the corresponding Stone space $\mathsf{Spec}(B)$ are the possible states of the environment. Now, the Boolean algebra is to be recovered as certain *decidable* functions on $\mathsf{Spec}(B)$,

[1] Email:richard.garner@mq.edu.au
[2] Email:{alyssa.renata19,n.wu}@imperial.ac.uk

and so this is formally enforced by imposing a topology on $\mathsf{Spec}(B)$ and taking the continuous functions as a proxy for the decidable ones. The Stone spaces characterize the spaces which may appear as the $\mathsf{Spec}(B)$ of some B, and so Spec is an equivalence between the category of Boolean algebras and Stone spaces.

Indeed, our exploration yields a duality. On the algebraic side we have the (ranked) monads on Set. In category theory, monads are understood as syntax-free algebraic theories [20], because while every algebraic theory induces a monad and vice versa, two algebraic theories with different generating operations (i.e. different syntax) may induce the same monad. We can replace algebraic theories by monads here because the environment is sensitive only to the computations underlying the programs, and not any particular choice of syntax. In subsections 2.1,2.2 and 2.3, we review the relevant aspects of monad theory.

On the geometric side, the environment associated to a monad T is a category $\mathsf{LB}T$ internal to "a category of spaces $\mathcal{S}$", which we call the *behaviour category* of T. Its objects are to be understood as states of the environment and morphisms as transitions between states. The internalization in spaces achieves the same effect of controlling for computability as in the classical Stone duality, but the first plot twist is that $\mathcal{S}$ is not the category Top of topological spaces. Rather, it is the category Loc of *locales*, also known as *pointless spaces* because they take as primitive the lattice of opens from the definition of topological spaces while omitting the points entirely. The reason for this twist is that there are monads whose topological space of states we found to be empty (example 3.7), and yet whose localic space of states is non-trivial (example 3.15)! Luckily, by restraining ourselves to finitary monads it does suffice to consider $\mathcal{S} = \mathsf{Top}$. The construction of the localic/topological behaviour category is addressed in section 3 (for the object/state space) and section 4 (for the morphism/transition space).

Now, before we can state our duality result, we must address the notion of morphism we take between our internal categories, wherein lies the second twist. On the monad side, we take a standard notion of monad morphism corresponding to interpretations of algebraic theories. On the geometric side however, the notion of morphism we need is *not* functors, but rather *retrofunctors* [22,1,3,4]. Roughly speaking, a retrofunctor $\mathbb{C} \rightsquigarrow \mathbb{D}$ takes objects from $\mathbb{C}$ to $\mathbb{D}$, but takes morphisms from $\mathbb{D}$ to $\mathbb{C}$. We can now state our first main contribution, though here lies the final twist: there is a contravariant *adjunction*, and not equivalence(!), between the category of ranked monads on Set and the category $\mathsf{LocRetro}$ of Loc-internal categories and retrofunctors, as well as its finitary cousin replacing internalization in Loc by Top.

Theorem 5.4 $\mathsf{LB} : \mathsf{Mnd}_r(\mathsf{Set}) \underset{\Gamma}{\overset{\mathsf{LB}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{LocRetro}^{\mathsf{op}} : \Gamma$ **Theorem 5.5** $\mathbb{B} : \mathsf{Mnd}_\omega(\mathsf{Set}) \underset{\Gamma_\omega}{\overset{\mathbb{B}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{TopRetro}^{\mathsf{op}} : \Gamma_\omega$

These are merely adjunctions because on the geometric side we assumed the environment is *deterministic*, but any algebraic theory with a commutative binary operation—such as the theory of non-deterministic finite choice—cannot be modeled by a deterministic environment [27], and so their corresponding environment under our adjunction is the trivial space, from which it is impossible to recover the original theory.

In general, for a ranked monad T, the recovered back-and-forth monad completes T with *prescience*: To each term $t \in TA$, there is a new operation $\bar{t}$ which—intuitively—performs t, keeps track of the result, and then rolls back the state of the environment to just before performing t. Monads admitting such *prescient computations* were characterised by the first-named author in [11,12] as the *cartesian closed* monads, i.e., those whose categories of Eilenberg-Moore algebras are cartesian closed. On the other side of the adjunction, the category we obtain from a monad satisfies what we term *ampleness* (definition 6.6), to use the terminology of C^*-algebraists [23, Definition 2.2.4]. The cartesian closed monads and the ample localic categories are precisely the fixpoints of our Stone adjunction, which thus restricts to an equivalence that constitutes the *Stone duality for monads* of the title, and our second main result.

Theorem 6.8 Adjunctions 5.4 and 5.5 restrict to $\mathsf{HUMnd}_\mathsf{r} \simeq \mathsf{AmpLocRetro}$ and $\mathsf{HUMnd}_\omega \simeq \mathsf{AmpTopRetro}$.

Now, the finitary version of the Stone duality obtains on the algebraic side the finitary cartesian closed monads, and on the geometric side the ample topological categories. From this, we find that this finitary Stone duality subsumes the classical Stone duality. on the algebraic side any Boolean algebra B induces a finitary monad T_B generated by binary operations of the form $\mathsf{if}\ b\ \mathsf{then}\ (-)\ \mathsf{else}\ (-)$ for each $b \in B$ [6]. On the geometric side, the space of objects for the category corresponding to T_B is the classical Stone dual $\mathsf{Spec}(B)$ of B, and the transitions consist of only identity morphisms, reflecting the read-only nature of computations in T_B. In fact, our story starts by inspecting the monad T_B towards the end of section 2, and curiously recovering $\mathsf{Spec}(B)$ via a construction on monads called the *terminal comodel*. This serves as a jumping off point for the rest of the paper.

Related and Future Work. The adjunction 5.5 is related to existing work on *interaction laws* [19]. First, adjunction 5.5 admits a detopologized version replacing $\mathsf{TopRetro}$ by the category Retro of small categories and retrofunctors. Retro is equivalent to the category of polynomial comonads [3,4], and composing this equivalence with our left adjoint functor $\mathbb{B}$, we recover the *cosemantics* functor of Garner [9]. As explained in that paper, the cosemantics comonad of a monad T is its *Sweedler dual* which is characterized as the universal comonad admitting an *interaction law* with T [19]. Here, the comonads represent environments for interacting with the monad, so the detopologized $\mathbb{B}T$ represents the universal environment for interacting with T. This brings us to our first direction for future work: can we explain the topologized $\mathbb{B}T$ or even $\mathsf{LB}T$ as a universal interacting comonad? This seems to demand a generalization of the Sweedler dual towards comonads on Top or Loc.

As we explained in the beginning, the behaviour category $\mathsf{LB}T$ should aid program reasoning. In future work, we hope to construct a logic for reasoning about programs in Moggi's monadic metalanguage [21]. We expect this logic to take the shape of propositional dynamic logic (PDL) [14] since $\mathsf{LB}T$ can be seen as a Kripke model whose propositions are interpreted as certain open sets of LB_0T and whose programs are (generated by) $T1$. The modality $[m]\varphi$ is interpreted as $(\!|m|\!)^{-1}\varphi$. Since we are interpreting the modalities in the monad $\Gamma\mathsf{LB}T$ completing T with propositions (the prescient computations), we don't require T to contain sufficiently many such propositions in the first place, lifting a constraint in previous works on monadic program logic such as Goncharov & Schröder [13].

Having brought up Goncharov & Schröder [13], we also ought to discuss their use of unary computations as propositions, whereas our propositions are generated by binary computations (see definition 3.12). This contrast seems to stem from their assumption that computations may fail to terminate, leading to an information ordering on computations à la domain theory, whereas we assume that computations always terminate. Since this assumption stems from a natural exploration of comodels, we hope in the future to explore the Stone-type duality that arises when we consider comodels *residual* [2,19,28] over the lifting monad $\{\perp\} + -$. Just as the global sections monad is a very fancy state monad, we expect the right adjoint of this duality to take monads of partial sections, i.e., fancy partial state monads.

It is also natural to consider generalizations beyond monads on Set. In this paper, many constructions explicitly talk about elements of monads, so an appropriate generalization will have to talk about morphisms in the Kleisli category instead. We can shed a preliminary light on this, based on the adjunction introduced by Cockett and the first-named author [8] between *restriction categories realized in* Loc and Loc-*internal partite categories*. Here, $\mathsf{LB}T$ is (or rather, generates) the partite internal category corresponding under this adjunction to a restriction category generated by the locale of states LB_0T. If we can construct LB_0T in element-free fashion, then we get a description of $\mathsf{LB}T$ which avoids talking about elements of T entirely.

Finally, the existence of a spectral duality for monads raise the interesting possibility of developing a *scheme theory of monads*. Recall the notion of a *scheme of rings* from algebraic geometry: these are locally ringed sheaves which are locally isomorphic to the spectrum of a commutative ring. The analogy here is between rings and (hyperaffine-unary) monads, with the spectrum of a ring (which is a sheaf) analogous to the source map of $\mathsf{LB}T$. At first approximation, a scheme of monads should then be a localic category S, which "locally resembles" $\mathsf{LB}T$ for some T. The idea is that in a scheme S, the monad (and hence the syntax) in play varies continuously over the base space S_0. This should allow for the modelling of effects whose syntax is not fixed, such as *local* state—it would be nice if "local state" = "locally a state monad".

2 Preliminaries, and the Classical Stone Duality from a Monadic Perspective

In subsections 2.1, 2.2, 2.3 and 2.4, we recall the basic theory of monads, their comodels, and the classification of comodels by the behaviour category, which is the prototype from which we build the topological and localic behaviour category. We then review the classical Stone duality from a comodel-theoretic perspective (subsection 2.5), using this to motivate our monadic Stone duality (subsection 2.6).

2.1 Monads

Monads encode computations, as first proposed by Moggi [21]. We use a definition which emphasizes this aspect of monads. Since this definition was popularized in functional programming, we also we also borrow some notation, specifically from the Haskell programming language.

Definition 2.1 A *monad* $(T, \gg=, \mathsf{return})$ on Set comprises:

(i) for each set A, a set TA of *computations*;

(ii) for each *value* $a \in A$, a *pure computation* $\mathsf{return}\, a \in TA$; and

(iii) for each set A, B a *composition* operation $\gg=\colon TA \times TB^A \to TB$.

These are required to satisfy, for each $t \in TA, a \in A, u \in TB^A, v \in TC^B$, the equations

$$\mathsf{return}\, a \gg= u = u(a) \qquad t \gg= \lambda a.\, \mathsf{return}\, a = t \qquad (t \gg= u) \gg= v = t \gg= (\lambda a.u(a) \gg= v).$$

A *monad map* $\gamma\colon T \to S$ comprises, for each set A, a function $\gamma_A\colon TA \to SA$ satisfying $\gamma(\mathsf{return}\, a) = \mathsf{return}\, a$ and $\gamma(t \gg= u) = \gamma(t) \gg= \lambda a.\gamma(u(a))$. We also introduce an extra piece of notation: if $t_1 \in TA$ and $t_2 \in TB$, then we write $t_1 \gg t_2$ for the composite $t_1 \gg= \lambda\_.t_2$ with the constant family on t_2.

Here, the set A of a computation $t \in TA$ is to be interpreted as a return type. A common way to specify monads is to specify a set Σ of generating operations op equipped with their return types A, which we denote by op/A for brevity. The monad F_Σ generated by such a set of operations has, for $F_\Sigma A$, the set of trees whose leaves are values of A and whose nodes are the generating operations with the branching arity specified by the return type. The operation $t \gg= u$ substitutes each leaf of t with label a by the tree $u(a)$, while $\mathsf{return}\, a$ is the leaf-only tree with label a.

One can furthermore specify generating equations, via a family of binary relations $\mathcal{E}_A$ over the set of trees $F_\Sigma A$. An *algebraic theory* $\mathbb{T} = [\Sigma|\mathcal{E}]$ consists of a set of generating operations O and equations R. The monad T generated by $\mathbb{T}$ is given by the quotient $TA = F_\Sigma A/{=_{\mathcal{E}_A}}$ where the family of equivalence relations $=_{\mathcal{E}_A}$ is generated by:

$$\frac{t_1\ \mathcal{E}_A\ t_2}{t_1 =_{\mathcal{E}_A} t_2} \qquad \frac{t_1 =_{\mathcal{E}_A} t_2 \quad \forall a \in A. u_1(a) =_{R_B} u_2(a)}{t_1 \gg= u =_{R_B} t_2 \gg= u_2} \qquad \frac{t \in TA}{t =_{\mathcal{E}_A} t} \qquad \frac{t_1 =_{\mathcal{E}_A} t_2}{t_2 =_{\mathcal{E}_A} t_1} \qquad \frac{t_1 =_{\mathcal{E}_A} t_2 \quad t_2 =_{\mathcal{E}_A} t_3}{t_1 =_{\mathcal{E}_A} t_3}$$

If $=_{\mathcal{E}_A}$ is the identity relation for all A then we say the theory and its corresponding monad is *free*.

Example 2.2 The monad of *binary input* is the free monad corresponding to the theory $[\mathsf{flip}/2|\emptyset]$, whose generating operation flip we can think of as sampling a binary digit, e.g. from a coin flip. A term $t \in TA$ of this monad is simply a binary tree with leaves labelled by elements of A. As a non-free example, the monad of *reversible input* is induced by further adding two new unary operations $\mathsf{unflip}_0/1$ and $\mathsf{unflip}_1/1$ along with the following equations expressing that the unflip operations are "inverses" of flip.

$$\left[\begin{array}{cc|c} \multicolumn{2}{c|}{\mathsf{flip}/2} & \multicolumn{2}{c}{\mathsf{flip}(\mathsf{unflip}_0(a), \mathsf{unflip}_1(a)) = \mathsf{return}\, a} \\ \mathsf{unflip}_0/1 & \mathsf{unflip}_1/1 & \mathsf{unflip}_0(\mathsf{flip}(a,b)) = \mathsf{return}\, a & \mathsf{unflip}_1(\mathsf{flip}(a,b)) = \mathsf{return}\, b \end{array}\right]$$

For this paper, we will want to distinguish operations based on the cardinality of their return types, and also consider monads whose operations are bounded by a certain cardinality; these are the *ranked* monads as follows. A special case is when all operations have finite return type; these are the *finitary* monads. After the following definition, any mention of monads will refer to ranked monads only.

Definition 2.3 An operation $t \in TA$ is *finitary* if there is some function $f\colon I \to A$ from a finite set I and $t' \in TI$ such that $t = t' \gg= \lambda i.\, \mathsf{return}\, f(i)$; if here we replace "finite" by "λ-small" for some regular cardinal λ, then we instead say that t is *λ-ary*. Now T itself is *finitary* if each of its operations is so, and is *ranked* if there is a regular cardinal λ for which all of its operations are λ-ary. We write $\mathsf{Mnd}_\omega(\mathsf{Set})$ (resp. $\mathsf{Mnd}_r(\mathsf{Set})$) for the category of finitary (resp. ranked) monads and monad maps.

2.2 Comodels

It is well-known that every monad is generated by some algebraic theory, and hence we can always interpret computations as trees. Such trees can be seen as *decision trees*, whose execution recursively proceeds by executing the topmost operation against some *environment* (See example 2.5 for a visualization). The

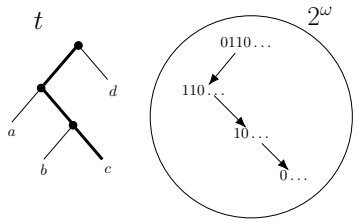
environment reciprocates by communicating which branch to walk down along, but is itself changed in the process. The execution then proceeds by executing the subtree found along the given branch, and so on. The execution terminates when the walk ends in a leaf, which becomes the return value of that execution. Environments are mathematically given by the notion of *comodel*, as first introduced in the context of computer science by Power and Shkaravska [26]. They are also called *stateful runners* by Uustalu [27], and as explained by Ahman and Bauer [2], we may also think of comodels as virtual machines for Σ.

Definition 2.4 Let T be a monad. A *T-comodel* is a pair $(W, (\!|-|\!))$ whose data comprises a set W of *states* along with *co-interpretations* $(\!|t|\!) \colon W \to A \times W$ of each computation $t \in TA$. These co-interpretations are required to respect $\gg\!=$ and return, in the sense that for any $t \in TA$, $u : A \to TB$, $a \in A$ and $w, w' \in W$,

$$(\!|\mathsf{return}\, a|\!)(w) = (a, w) \text{ and } (\!|t \gg\!= u|\!)(w) = (\!|u(a)|\!)(w') \text{ where } (a, w') = (\!|t|\!)(w).$$

A *comodel map* is a function $W \to W'$ which preserves each co-interpretation, i.e., for each $t \in TA$, $(\!|t|\!)^{W'}(h(w)) = h((\!|t|\!)^{W}(w))$. We write Comod_T for the category of T-comodels.

Example 2.5 An example comodel for the binary input monad is the set 2^ω of infinite binary streams. Because co-interpretations respect $\gg\!=$, the co-interpretations for a monad generated by a theory $[\Sigma|\mathcal{E}]$ is determined by the co-intepretations of the generating operations Σ. In this case, it means that we need only specify a map $(\!|\mathsf{flip}|\!) : 2^\omega \to 2 \times 2^\omega$ for which we take the map $\beta \mapsto (\mathsf{head}(\beta), \mathsf{tail}(\beta))$. Then the co-interpretation of a term occurs by consuming one digit to walk down one operation at a time, as the righthand diagram demonstrates in co-interpreting $t = \mathsf{flip}(\mathsf{flip}(a, \mathsf{flip}(b, c)), d)$ against state $\beta = 0110\ldots$. The co-interpretation finishes when it reaches the leaf c, leaving the environment in state $\beta' = 0\ldots$, so $(\!|t|\!)(\beta) = (c, \beta')$.



The same comodel works for reversible binary input, where the unflip operations put the corresponding digit back on top of the given stream, i.e., $(\!|\mathsf{unflip}_0|\!) \colon \beta \mapsto 0 : \beta$ and $(\!|\mathsf{unflip}_1|\!) \colon \beta \mapsto 1 : \beta$. Clearly, any comodel of reversible input is also a comodel of binary input, but consider the set with two states which we shall call $(01)^*$ and $(10)^*$, intuitively the stream of repeating 01s and 10s respectively. This admits a cointerpretation $(\!|\mathsf{flip}|\!)((01)^*) = (0, (10)^*)$ and $(\!|\mathsf{flip}|\!)((10)^*) = (1, (01)^*)$, but this cannot be extended to the unflip operations in a way that would respect the generating equations, since this would require adding new states such as $0(01)^*$ to cointerpret $(\!|\mathsf{unflip}_0|\!)((01)^*)$.

2.3 The Terminal Comodel

Hopefully, the reader has gathered enough intuition on comodels by now to see that they look very much like transition systems. Indeed, comodels are coalgebras, so it is instructive to look at the terminal object which encodes the observable behaviours of coalgebraic states. In the case of comodels, the observable behaviour of some state is to execute any given computation $t \in TA$ down to a return value $a \in A$, so the elements of the terminal comodel are given by maps $TA \to A$ (natural over A), admitting a certain *admissibility* condition [9].

Definition 2.6 (Admissible behaviours) A *T-behaviour* β simply consists of an assigment $\beta_A : TA \to A$ for each set A. Such a behaviour β is *admissible* if for any $t \in TA$, $a \in A$ and $u \colon A \to TB$, we have $\beta(t \gg\!= u) = \beta(t \gg u(\beta(t)))$ and $\beta(t \gg \mathsf{return}\, a) = a$. Note that the latter equation is equivalent to naturality of $\beta : T \to \mathsf{id}_{\mathsf{Set}}$ in the presence of the former. The set of admissible behaviours $\mathbb{B}_0 T$ admits a comodel structure with $(\!|t|\!)(\beta) = (\beta(t), \partial_t \beta)$ where the next behaviour $\partial_t \beta$ is defined by $\partial_t \beta \colon t' \mapsto \beta(t \gg t')$. This makes $\mathbb{B}_0 T$ the terminal comodel, where for any other comodel W, the unique comodel map $W \to \mathbb{B}_0 T$ sends $w \in W$ to the behaviour $\beta_w \colon t \in TA \mapsto \pi_A((\!|t|\!)(w))$.

Unless otherwise indicated, every behaviour we consider from now on will be admissible.

Example 2.7 For our running example of the binary input monad, the comodel 2^ω is in fact (isomorphic to) the terminal comodel. This is because by admissibility of any behaviour β, the execution of $\beta(t)$ for any tree t is determined by the values of $\beta(\mathsf{flip}), \beta(\mathsf{flip} \gg \mathsf{flip}), \ldots \beta(\mathsf{flip}^k) \ldots$, which determines a map $\omega \to 2$. For example, for the t in example 2.5 $\beta(t)$ is determined by $\beta(\mathsf{flip}) = 0$, $\beta(\mathsf{flip}^2) = 1$ and $\beta(\mathsf{flip}^3) = 1$. Since

the binary input monad is free, any such map $\omega \to 2$ determines an admissible behaviour, and hence we obtain our isomorphism. This argument also works for reversible binary input, since β has no choice to make for computations involving the unflip operations.

Any comodel W admits a unique map $!\colon W \to \mathbb{B}_0 T$, which by taking inverse images determines a map $!^{-1} : \mathbb{B}_0 T \to \mathsf{Set}$. On the other hand, not just any family $P : \mathbb{B}_0 T \to \mathsf{Set}$ can induce a comodel, but the first-named author [9] showed that we can equip $\mathbb{B}_0 T$ with morphisms $\mathbb{B}_1 T$ to obtain a *behaviour category* $\mathbb{B}T$ such that covariant presheaf functors $P\colon \mathbb{B}T \to \mathsf{Set}$ correspond to comodels by taking the sum of the sets $P(\beta)$ over all the behaviours β. We now review the construction of the behaviour category.

2.4 The Behaviour Category

For a free monad T, a T-behaviour β produces, from each term $t \in TA$, a sequence of operations called the *trace*, which it had to evaluate to run t down to its return value. These traces are what we take as the morphisms of the behaviour category. In fact, the return value itself does not matter for computing the trace, i.e., $t \in TA$ and $(t \gg \mathsf{return}) \in T1$ have the same trace, so it is enough to consider the trace of $T1$-terms only. Now, an arbitrary monad T may have more than one plausible set of generating operations, so we want a definition of traces which is agnostic to any particular choice of generators. We do this by defining trace-equivalence and then recover the traces as the trace-equivalence classes.

Definition 2.8 (β-equivalence, behaviour category) Let β be a T-behaviour. The relation $\sim_\beta$ on $T1$ is the least equivalence relation such that $(t \gg\!= u) \sim_\beta (t \gg u(\beta(t)))$ for any $t \in TA$ and $u\colon A \to T1$. Write $[t]_\beta$ for the $\sim_\beta$-equivalence class of t. The *behaviour category* $\mathbb{B}T$ has as objects T-behaviours, and as morphisms pairs of the form $(\beta, [t]_\beta)$ but which we will simply write as $[t]_\beta$. The morphism $[t]_\beta$ has source β and target $\partial_t \beta$. The identity id_β is $[\mathsf{return}]_\beta$, while the composite $[t]_\beta; [s]_{\partial_t \beta}$ is $[t \gg s]_\beta$.

Theorem 2.9 $\mathsf{Comod}_T \simeq [\mathbb{B}T, \mathsf{Set}]$.

Example 2.10 Let T be the monad of binary input. The trace of $t \in T1$ at β is the number of flips traversed by β when running down t, so a morphism in $\mathbb{B}T$ has the form $n\colon W \to \partial^n W$ for some $n \in \mathbb{N}$.

Example 2.11 For any set S, there is a *state monad* $T = (- \times S)^S$ whose computations are functions which inspect and update the current state, along with using this information to produce a return value. Its behaviours are in bijection with the set of states S, and actually so are the traces $T1/_{\sim_s}$ over each $s \in S$. To understand why the latter is true, observe that the unary terms are just state-update functions $S \to S$. Now, intuitively, two such unary computations are equivalent if they make the same update at state s. Indeed, two unary terms $t_1, t_2 \in T(1) = S^S$ are s-equivalent iff $t_1(s) = t_2(s)$. So $\mathbb{B}T$ is the indiscrete (or chaotic) category with object-set S. We refer to [9] for more examples.

2.5 The Classical Stone Duality, from a Comodel-theoretic Perspective

Recall that a Boolean algebra is a bounded distributive lattice $(B, \wedge, \vee, \top, \bot)$ such that every element $b \in B$ admits a complement, i.e., a (necessarily unique) element $\neg b$ such that $b \vee \neg b = \top$ and $b \wedge \neg b = \bot$. From a logical perspective, we think of the elements of B as propositions about the world. Any Boolean algebra induces a *monad of partitions* [6,11], which we now define "out of thin-air". However, this monad does admit an equational presentation which we actually use later in this paper (see remark 4.4).

Definition 2.12 Let B be a Boolean algebra. For a set A, an *A-partition* of B is a function $t : A \to B$ such that p is (i) *finitely supported*, i.e., for all but finitely many $a \in A$, $t(a) = \bot$; (ii) *pairwise disjoint*, i.e., $t(a) \wedge t(a') = \bot$ for $a \neq a'$; and (iii) *jointly covering*, i.e., $\bigvee \{\, t(a) \neq \bot \mid a \in A \,\} = \top$. The monad of partitions T_B takes $T_B A$ to be the set of A-partitions, with $(\mathsf{return}\, a)(a) = \top$ and $\bot$ otherwise. The composite $t \gg\!= u$ for $t \in TA$ and $u : A \to TC$ is the partition $(t \gg\!= u)(c) = \bigvee_{a \in A} t(a) \wedge u(a)(c)$.

While this monad seems quite complicated, it is generated by $T_B 2$ which is in fact isomorphic to the Boolean algebra B. To see why, notice that a binary partition t determines some $b := t(1)$, and then the conditions demand that $t(0) = \neg b$. We can view such a t as a program expression $\mathsf{if}\ b\ \mathsf{then}\ 1\ \mathsf{else}\ 0$. In this case, executing such an expression amounts to checking whether b holds in the world, and then

returning the appropriate value. The observable behaviour of the world then is just a specification of which propositions hold, subject to some reasonable conditions. Such specifications are called *ultrafilters*.

Proposition 2.13 *Let B be a Boolean algebra. The terminal comodel $\mathbb{B}_0 T_B$ is isomorphic to the set of* ultrafilters *for B, namely those subsets $\mathfrak{p} \subset B$ that satisfy:* (i) nontriviality*, i.e., $\top \in \mathfrak{p}$;* (ii) completeness*, i.e., $a \vee b \in \mathfrak{p}$ implies $a \in \mathfrak{p}$ or $b \in \mathfrak{p}$; and* (iii) consistency*, i.e., $\neg a \in \mathfrak{p}$ implies $a \notin \mathfrak{p}$.*

Now, the much-celebrated Stone duality theorem says that we can actually recover the Boolean algebra B *if* we additionally induce a topology on the set of ultrafilters. The topological spaces that arise in this way are called the *Stone spaces* (definition 3.5), and by Stone duality there is a dual equivalence $\mathsf{BA} \simeq \mathsf{Stone}^{\mathsf{op}}$. In combination with the above proposition, the monads of the form T_B are therefore determined by their terminal comodel equipped with an appropriate topology. Therefore it is natural to ask: can we generalize Stone duality to recover any monad T from its terminal comodel $\mathbb{B}_0 T$?

2.6 Towards a Generalized Stone Duality for Monads

Since the terminal comodel interprets each computation t as a map $(\!|t|\!) \colon \mathbb{B}_0 T \to A \times \mathbb{B}_0 T$, it is natural as a naive attempt to try and recover the monad T as the *state monad* $(- \times \mathbb{B}_0 T)^{\mathbb{B}_0 T}$. But there are two orthogonal problems to consider.

First, we already know from the classical Stone duality that we ought to put some kind of topology on $\mathbb{B}_0 T$, which we have yet to do. Actually, there is a computational explanation for using topology: there are too many functions $\mathbb{B}_0 T \to A \times \mathbb{B}_0 T$, not all of which are actually computable. Consider the binary input or even reversible input monad for example. There is no binary tree whose cointerpretation is a map $2^\omega \to \{\, \mathsf{yes}, \mathsf{no} \,\} \times 2^\omega$ assigning $\mathsf{repeat}(10) \mapsto \mathsf{yes}$, and no for any other stream, because this requires inspecting infinitely many digits of the stream. Putting a topology, and asking for only continuous functions, acts as a proxy for demanding computability [3]. As we will see later, the topology we put on $\mathbb{B}_0 T$ coincides with the cantor space topology when T is binary input, which rules out the function we just described. In section 3 we address this topologization for arbitrary monads, capturing the Stone topology as a special case. It turns out there that topology is only sufficient to capture finitary monads, for reasons owing to size—hence our sensitivity to cardinality when introducing monads earlier. To better capture ranked monads, we need to consider replacing topological spaces by an alternative notion of space called *locales*.

Second, both the binary input monad and the reversible input monad induce the same terminal comodel, so we can't yet distinguish the two. Still, they have different categories of comodels, as we saw in example 2.5. The solution for this is to consider not just $\mathbb{B}_0 T$, but the behaviour category $\mathbb{B}T$; since it classifies categories of comodels, it must be able to tell the difference between binary input and reversible input. We have already seen in example 2.10 that the morphisms of the behaviour category for binary input are natural numbers specifying how many digits of the stream to consume. For reversible input, we have said the unflip operations are inverses for flip, and indeed the behaviour category for reversible input is a *groupoid* whose morphisms are, in addition to the natural numbers, non-empty lists of binary digits specifying digits to put back onto the stream. Amalgamating our proposed solutions to the two problems, we must not only topologize (or rather, localify) $\mathbb{B}_0 T$, but also $\mathbb{B}_1 T$, and this is exactly what we address in section 4 to obtain the *localic behaviour category* $\mathsf{LB}T$ as the Stone dual of T. The remaining two sections section 5 and section 6 take the story to its logical conclusion by constructing the dual equivalence of categories that becomes our Stone duality for monads.

3 Topological & Localic Comodels

3.1 Topological Comodels

In this section, our goal is to topologize the terminal comodel. We do this by rediscovering the terminal object in the category $\mathsf{Comod}_T(\mathsf{Top})$ of comodels valued in Top, the category of topological spaces. For this reason, we need to define comodels valued in other categories beyond Set. To support a definition of comodels, the base category $\mathcal{C}$ must have *copowers*, which means that for any $C \in \mathcal{C}$ and any set A, the

[3] This is similar to Scott's model of the untyped lambda-calculus: the only set X isomorphic to X^X is $X = 1$, but there are non-trivial *spaces* X for which such isomorphisms exist.

A-fold coproduct $A \cdot C$ exists. For example, both Set and Top have copowers given by the A-fold disjoint sum $\coprod_{a \in A} C$. We will denote the inclusion maps by $\upsilon_a \colon C \to A \cdot C$, and the codiagonal by $\pi_C \colon A \cdot C \to C$.

Definition 3.1 [9, Definition 2.15] Let T be a monad and $\mathcal{C}$ a category with copowers. A *comodel* of T in $\mathcal{C}$ is a pair $(W, (\!|-|\!))$ whose data comprises an object $W \in \mathsf{ob}\,\mathcal{C}$ and *co-interpretations* $(\!|t|\!) \colon W \to A \cdot W$ for each computation $t \in TA$. If we extend this to a cointerpretation $(\!|u|\!) \colon A \cdot W \to B \cdot W$ of each $u \colon A \to TB$ via $(\!|u|\!) := [(\!|u(a)|\!)]_{a \in A}$, then the comodel axioms require that $(\!|t \gg\!= u|\!) = (\!|u|\!) \circ (\!|t|\!)$ and $(\!|\mathsf{return}\, a|\!) = \upsilon_a$. A *comodel map* $(W, (\!|-|\!)_W) \to (W', (\!|-|\!)_{W'})$ is a map $h \colon W \to W'$ which preserves each co-interpretation, i.e., for each $t \in TA$, we have $(\!|t|\!) \circ h = (A \cdot h) \circ (\!|t|\!)$. We write $\mathsf{Comod}_T(\mathcal{C})$ for the category of T-comodels in $\mathcal{C}$.

This recovers definition 2.4 with $\mathsf{Comod}_T = \mathsf{Comod}_T(\mathsf{Set})$. Clearly, any Top-comodel is a Set-comodel, yielding a forgetful functor $U \colon \mathsf{Comod}_T(\mathsf{Top}) \to \mathsf{Comod}_T(\mathsf{Set})$. On the other hand, there is also a coarsest topology on any Set-comodel making it a Top-comodel [10]:

Definition 3.2 (Operational Topology) Let W be a T-comodel in Set. The *operational topology* on W is generated by sub-basic open sets $[t \mapsto a] := \{\, w \in W \mid (\!|t|\!)(w) = (a, w') \text{ for some } w' \,\}$ for $t \in TA$ and $a \in A$.

Proposition 3.3 *The assignment which endows a comodel with its operational topology yields a right adjoint* $O \colon \mathsf{Comod}_T(\mathsf{Set}) \to \mathsf{Comod}_T(\mathsf{Top})$ *to* $U \colon \mathsf{Comod}_T(\mathsf{Top}) \to \mathsf{Comod}_T(\mathsf{Set})$.

Proof. Let $\mathcal{W}$ be a Top-comodel and W a Set-comodel. Then any Set-comodel morphism $f \colon U\mathcal{W} \to W$ is a continuous function $f \colon \mathcal{W} \to OW$ since $f^{-1}[t \mapsto a]_W = [t \mapsto a]_{\mathcal{W}} = (\!|t|\!)^{-1}_{\mathcal{W}}(\{a\} \times \mathcal{W})$. □

Since right adjoints preserve limits—so in particular terminal objects—we can conclude that $\mathbb{B}_0 T$ equipped with the operational topology is the terminal Top-comodel.

Example 3.4 (The Cantor Space) Consider the monad of binary input and its terminal topological comodel. In this case, we can choose an even smaller set of sub-basic opens, namely those of the form $[\mathfrak{b}] = \{\, \beta \mid \beta \sqsupseteq \mathfrak{b} \,\}$ of those streams extending some finite sequence $\mathfrak{b}$ of binary digits. This space is better known as the *Cantor space*.

Now, if $T = T_B$ is the partitions monad over a Boolean algebra B, since this monad is generated by the binary partitions in correspondence with elements $b \in B$, the operational topology on $\mathbb{B}_0 T$ is generated by open sets of the form $[b \mapsto 1]$ (and also $[b \mapsto 0]$, but those are recoverable as $[\neg b \mapsto 1]$), which we can simply write as $[b]$. Under the correspondence of proposition 2.13, we find that the generating open sets are of the form $[b] = \{\, \mathfrak{p} \mid b \in \mathfrak{p} \,\}$, and this is precisely the Stone topology on the set of ultrafilters. The spaces that arise in this way are quite well-behaved, and are known as the *Stone spaces*.

Definition 3.5 A *Stone space* X is a topological space X which is compact, Hausdorff and *zero-dimensional*, i.e., admits a basis of clopen sets.

For any monad T, $\mathbb{B}_0 T$ is actually always Hausdorff and by construction admits a basis of clopen sets (each $[t \mapsto a]$ is clopen), but the compactness condition requires T to be finitary, though we obtain a proof of this as a corollary of propositions 3.19 and 3.20 from later in this section.

Proposition 3.6 *Let* T *be a finitary monad. Then* $\mathbb{B}_0 T$ *is a Stone space.*

As for infinitary monads, the following example demonstrates that the terminal topological comodel for certain infinitary monads T are poorly behaved, even if they admit a deterministic and non-failing notion of environment.

Example 3.7 Consider the monad T corresponding to the following theory.

$$\left[\begin{array}{l|c} \forall x \in \mathbb{R}, & \mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_x \lambda m.\, \mathsf{return}(n,m) = \mathsf{get}_x \lambda n.\, \mathsf{return}(n,n) \qquad \mathsf{get}_x \gg \mathsf{return}\, a = \mathsf{return}\, a \\ \mathsf{get}_x/\mathbb{N} & \mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_y \gg\!= \lambda m.\, \mathsf{return}(n,m) = \mathsf{get}_y \gg\!= \lambda m.\mathsf{get}_x \gg\!= \lambda n.(n,m) \\ & \mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_y \gg\!= \lambda m.\, \mathsf{return}(n,m) = \mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_y \gg\!= \lambda m.f(n,m) \\ & \text{for all } x \neq y \in \mathbb{R} \text{ and } f \colon \mathbb{N} \times \mathbb{N} \to \mathbb{N} \times \mathbb{N} \text{ satisfying } \forall n \neq m \in \mathbb{N}. f(n,m) = (n,m) \end{array}\right]$$

The first three equations express that get_x fetches the value of a natural number from some memory location x, without changing the stored value, i.e., they express the theory of *read-only state*. The behaviours correspond to functions $\mathbb{R} \to \mathbb{N}$ each of which express some memory configuration. The fourth equation is our non-standard addition, expressing that two distinct memory locations cannot contain the same value. This makes $\mathbb{B}_0 T$ empty since the admissible behaviours in this case corresponds to injective memory configurations $\mathbb{R} \rightarrowtail \mathbb{N}$, of which there are (famously) none.

However, a term in this signature of operations is well-founded, which is to say any particular execution of a term in T can only query finitely many memory locations. So, from the program's perspective, it can never be sure that it is *not* in a non-injective state configuration, since it needs to query uncountably-many memory locations to make a pigeonhole principle argument. We can think of this as having an uncountable virtual address space over countably many physical memory cells—the moment you try to query more locations than there are actual cells, the program forcibly halts.

Clearly, we cannot recover the monad T of injective read-only state from the empty $\mathbb{B}_0 T$, even though it is in some sense possible to reason consistently about such state configurations. In fact, there is a *non-trivial "space" of injective functions* $\mathbb{R} \rightarrowtail \mathbb{N}$ [18, Example C1.2.9], but this requires us to change what we mean by "space", choosing a notion of space that emphasizes the opens, and de-emphasizes the points. Such "spaces" are called *locales*, whose theory we now review, before computing the terminal localic comodel.

3.2 Frames & Locales

Locales are spaces where we emphasize the opens, which in a topological space forms a poset (under $\subset$) with infinite joins (unions $\bigcup$) and finite meets (intersection $\cap$) that distribute. Locales are defined more abstractly as any such poset, but there is some subtlety on the morphisms.

Definition 3.8 A *frame* is a poset with infinite joins and finite meets satisfying the infinite distributive law $x \wedge (\bigvee_i y_i) = \bigvee_i (x \wedge y_i)$. We write Frm for the category of frames and frame homomorphisms, i.e., monotone maps which preserve infinite joins and finite meets. A *locale* is simply a frame, but the category of locales is $\mathsf{Loc} := \mathsf{Frm}^{\mathsf{op}}$.

The reason for this flipping of morphisms is that frame homomorphisms more closely resemble the inverse image map f^{-1} associated to a continuous function $f : X \to Y$, which by continuity takes open sets of Y to open sets of X, i.e., continuous functions go in the opposite direction to their induced frame homomorphisms. In fact, we have a functor $\mathcal{O} \colon \mathsf{Top} \to \mathsf{Loc}$ which sends X to its frame of open sets $\mathcal{O}(X)$, and a continuous map $f \colon X \to Y$ to $f^{-1} \colon \mathcal{O}(Y) \to \mathcal{O}(X)$. We imagine a general locale to be the lattice of opens of some space, and the data of a continuous map to be given by the inverse image map. To sustain this fantasy, we overload notation by also writing $\mathcal{O} \colon \mathsf{Loc}^{\mathsf{op}} \to \mathsf{Frm}$ for the identity functor, writing its action on morphisms as $\mathcal{O}(f) := f^{-1}$, and calling elements $u \in \mathcal{O}(L)$ *opens* of L.

Definition 3.9 Any locale L induces a topological space $\mathsf{pt}(L)$ with set of points the locale morphisms $x \colon 1 \to L$ from the terminal locale 1 and open sets given by $[u] = \{\, x \mid x^{-1} u = \top \,\}$ for $u \in \mathcal{O}(L)$. This yields a functor $\mathsf{pt} \colon \mathsf{Loc} \to \mathsf{Top}$, right adjoint to $\mathcal{O}$. A locale L at which the adjunction counit is invertible is called *spatial*, while a space at which the unit is invertible is called *sober*: thus, spatial locales and sober spaces form equivalent categories.

We make use of the fact—see [16, II 2.11]—that frames may be constructed by generators and relations.

Example 3.10 (Copowers in Loc) The (frame of opens of the) copower locale $A \cdot L$ can be presented by generating opens of the form $\langle a \mapsto u \rangle$ for $a \in A$ and $u \in \mathcal{O}(L)$, subject to the equations (1) $\bigvee_i \langle a \mapsto u_i \rangle = \langle a \mapsto \bigvee_i u_i \rangle$; (2) $\bigwedge_i^k \langle a \mapsto u_i \rangle = \langle a \mapsto \bigwedge_i^k u_i \rangle$; and (3) $\langle a \mapsto u \rangle \wedge \langle a' \mapsto v \rangle = \bot$ for $a \neq a'$. These equations imply that every open is uniquely of the form $\bigvee_{a \in A} \langle a \mapsto u_a \rangle$—which is consistent with the fact that A-fold copowers in Loc correspond to A-fold products in Frm.

Since Loc has copowers, we may also consider $\mathsf{Comod}_T(\mathsf{Loc})$, and the adjunction between Top and Loc lifts to their corresponding categories of T-comodels as the following proposition establishes.

Proposition 3.11 *The adjunction* $\mathcal{O} \dashv \mathsf{pt} \colon \mathsf{Loc} \to \mathsf{Top}$ *lifts to* $\mathcal{O} \dashv \mathsf{pt} \colon \mathsf{Comod}_T(\mathsf{Loc}) \to \mathsf{Comod}_T(\mathsf{Top})$.

Proof sketch. Since comodels are defined in terms of copowers, it suffices to verify that $\mathcal{O}$ and pt preserve copowers. Since $\mathcal{O}$ is a left adjoint, it preserves colimits and in particular copowers. As for pt, this is one of its standard properties, which follows from fact that the terminal locale 1 is connected, so that homming out of it preserves copowers. □

3.3 The Terminal Localic Comodel

By proposition 3.11, we see that the terminal topological comodel has to be the spatialization of the terminal localic comodel. But by proposition 3.3, the terminal topological comodel is the set of behaviours, equipped with the operational topology. This gives us an idea of what the terminal localic comodel looks like.

Definition 3.12 Let T be a monad on Set. The *behaviour locale* LB_0T is generated by opens $[b]$ for each $b \in T2$, subject to the following equations for all $t \in TA, u : A \to TB, a \neq a' \in A$ and $b \in B$.

$$[t \mapsto a] \wedge [t \mapsto a'] = \bot \qquad (\mathsf{LB}_0\text{-}\bot) \qquad\qquad [t \gg \mathsf{return}\, a \mapsto a] = \top \qquad (\mathsf{LB}_0\text{-}\eta)$$

$$[t \gg\!= u \mapsto b] = \bigvee_{a \in A} [t \mapsto a] \wedge [t \gg u(a) \mapsto b] \qquad (\mathsf{LB}_0\text{-}\mu)$$

Here we write $[t \mapsto a]$ as shorthand for $[t \gg\!= \lambda a'.\delta_a(a')]$ with $\delta_a(a') = 1$ when $a = a'$ and 0 otherwise.

Proposition 3.13 *Let T be a monad on Set. Then the following equations hold in LB_0T:*

$$\bigvee_{a \in A} [t \mapsto a] = \top \qquad [t \gg \mathsf{return}\, a \mapsto a'] = \bot \qquad [t \mapsto a] \wedge [t \gg\!= u \mapsto b] = [t \mapsto a] \wedge [t \gg u(a) \mapsto b],$$

where $t \in TA$, $a \neq a' \in A$, $u \colon A \to TB$ and $b \in B$.

In fact, axiom (LB_0-μ) can equivalently be replaced by a combination of the first and third equations of proposition 3.13. We chose axioms (LB_0-μ) and (LB_0-η) because of their resemblance to the admissibility condition of T-behaviours (definition 2.6). In any case, the axioms can be "discovered" as the necessary conditions for proving the universal property of LB_0T as the terminal localic comodel, as in the following proposition.

Proposition 3.14 *LB_0T is the terminal localic comodel with cointerpretation $(\!|t|\!) \colon \mathsf{LB}_0T \to A \cdot \mathsf{LB}_0T$ given by: $(\!|t|\!)^{-1} \colon \langle a_0 \mapsto [t_0] \rangle \mapsto [t \mapsto a_0] \wedge [t \gg t_0]$.*

Now by proposition 3.11 we conclude that the spatialization of behaviour locale LB_0T is the topological space $\mathbb{B}_0T$. This says that LB_0T contains more information than $\mathbb{B}_0T$, so we ought to check that it contains *strictly* more by making sure that we have gained something with respect to example 3.7.

Example 3.15 Consider again the monad T of injective read-only state from example 3.7. Its behaviour locale LB_0T, which in this instance is the the locale of injective functions $\mathbb{R} \rightarrowtail \mathbb{N}$ from [18, Example C1.2.9]. The underlying frame of this locale is generated by opens $\langle x \mapsto n \rangle$ for $x \in \mathbb{R}$ and $n \in \mathbb{N}$, under the requirement that, for any $x \neq y$ and $m \neq n$, the following equations holds:

$$\bigvee_{x \in \mathbb{R}} \langle x \mapsto n \rangle = \top \qquad \langle x \mapsto n \rangle \wedge \langle x \mapsto m \rangle = \bot \qquad \langle x \mapsto n \rangle \wedge \langle y \mapsto n \rangle = \bot.$$

To see why, notice that by repeated application of axiom (LB_0-μ) the behaviour locale for this monad LB_0T can instead be generated by opens of the form $[\mathsf{get}_x \mapsto n]$, which of course coincide with the generating opens $\langle x \mapsto n \rangle$ above. Then it is a matter of proving that the equations of the behaviour locale reduce, in this instance, to the equations above—this proof is documented in appendix A.3.

3.4 Strongly Zero-dimensional Locales and Grothendieck Boolean Algebras

In proposition 3.6 the terminal topological comodel is shown to be a Stone space whenever T is finitary. We now want to state and prove a generalization of this proposition for infinitary monads, dropping the

compactness requirement. In the absence of compactness, the zero-dimensionality condition of Stone spaces needs to be infinitarily strengthened, obtaining the *strongly zero-dimensional* locales of Johnstone [15], also called *ultraparacompact* by Van Name [29].

Definition 3.16 [29] Let L be a locale. An open $u \in \mathcal{O}(L)$ is *complemented* if it so in the usual lattice-theoretic sense: thus, there is some $v \in \mathcal{O}(L)$ with $u \wedge v = \bot$ and $u \vee v = \top$. The set $\mathfrak{B}(L)$ of complemented opens of L inherits finite meets and joins from L, and so is a Boolean algebra. We say that L is *zero-dimensional* if $u \in \mathcal{O}(L)$ is the join of the complemented opens below it.

A *cover* of L is a subset $J \subseteq \mathcal{O}(L)$ such that $\bigvee j = \top$. A cover J *refines* J' if for every $u \in J$ there is $u' \in J'$ such that $u \leq u'$. An *extended partition* P is a pairwise disjoint cover, i.e., $u \wedge v = \bot$ for any $u \neq v \in P$. A *partition* P is an extended partition which does not contain $\bot$, and any extended partition P induces a partition $P^- = P \setminus \{ \bot \}$. A zero-dimensional locale L is *strongly zero-dimensional* or *ultraparacompact* if every open cover is refined by a partition.

Since ultraparacompact locales generalize Stone spaces, there should be a corresponding generalization of Stone duality. On the algebraic side, the corresponding generalization is to consider the *Grothendieck Boolean algebras*. The notion is originally due to [29] but our nomenclature follows [11, Definition 3.6].

Definition 3.17 A *Grothendieck Boolean algebra* $B_\mathcal{J}$ is a Boolean algebra equipped with a *strongly zero-dimensional topology*, i.e., a collection $\mathcal{J}$ of partitions for B such that

(i) $\mathcal{J}$ contains every finite partition;
(ii) if $P \in \mathcal{J}$ and $Q_b \in \mathcal{J}$ for each $b \in P$, then $P;Q := \{ b \wedge c \mid b \in P, c \in Q_b \}^- \in \mathcal{J}$ also;
(iii) if $P \in \mathcal{J}$ and $f\colon P \to I$ is a surjective function, then each $\bigvee f^{-1}(i)$ exists in B and $f^{-1}P := \{ \bigvee f^{-1}(i) \mid i \in I \} \in \mathcal{J}$.

Theorem 3.18 [29, Theorem 24] *The category of ultraparacompact locales is dually equivalent to the category of Grothendieck Boolean algebras.*

Proof. We sketch just the constructions. An ultraparacompact locale L induces a Boolean algebra $\mathfrak{B}(L)$ with strongly zero-dimensional topology given by the partitions of L (the opens in a partition are necessarily complemented). On the other hand, a Grothendieck Boolean algebra $B_\mathcal{J}$ generates a locale of $\mathcal{J}$-closed ideals in the usual way (as explained by Vickers [30] or Johnstone [16, II 2.11]). □

We are now in the position to show that LB_0T is ultraparacompact. The intuition here is that the equations defining LB_0T can be re-expressed as generating equations for a Boolean algebra plus a strongly zero-dimensional topology on that algebra. The topology is essentially generated by the first equation of proposition 3.13). When T is finitary, this is expressible as a finite disjunction, so the topology is the topology of finite partitions, and hence LB_0T is compact because every open cover in LB_0T is by construction refinable by a finite partition, which yields a finite subcover. Hence we have the following proposition, whose proof details can be found in appendix A.4.

Proposition 3.19 *Let T be a monad. Then LB_0T is ultraparacompact. Moreover, if T is finitary, then LB_0T is compact.*

Now, we arrived at the notion of ultraparacompactness by eliminating compactness, so in the presence of compactness, ultraparacompactness amounts to being a Stone space. So in particular, while example 3.15 shows that we need the full force of locales for infinitary monads, it suffices to consider the Stone space $\mathbb{B}_0T$ for finitary monads.

Proposition 3.20 *Let L be a compact locale. Then L is ultraparacompact iff L is spatial and $\mathsf{pt}\,L$ is a Stone space, so in particular $\mathsf{pt}\,\mathsf{LB}_0T \cong \mathbb{B}_0T$ is a Stone space (thus also proving proposition 3.6).*

Remark 3.21 Every Boolean algebra can be regarded as a Grothendieck Boolean algebra under the topology of finite partitions. In that case, the dual locale is spatial, and as a topological space is precisely the classical Stone dual. In this way, the equivalence of 3.18 subsumes the usual Stone duality.

4 The Localic Behaviour Category

In this section, we construct the localic behaviour category $\mathsf{LB}T$. Following the construction of the behaviour category, the locale of objects LB_0T can be characterized as the terminal localic comodel. Hence, the main battle in this section is defining the *locale of morphisms* LB_1T.

Consider the usual behaviour category to begin with. For a monad T, any operation $t \in TA$ induces a section $\eta(t)\colon \mathbb{B}_0T \to A \cdot \mathbb{B}_1T$ of the behaviour category's source map $\sigma : \mathbb{B}_1T \to \mathbb{B}_0T$. This suggests we should recover a monad from the behaviour category by taking such sections. But as we have already explained in subsection 2.5, we must consider only the *continuous sections*.

Now, we can flip this argument on its head: we are trying to construct a map $\sigma : \mathsf{LB}_1T \to \mathsf{LB}_0T$ without knowing the domain space. The only criterion we have is that we know what continuous sections it should admit, namely the ones induced by $\eta(t)$ for all $t \in TA$. In this case, it is well-known that the problem admits a universal solution: first, describe a sheaf consisting of the "sections" which you want to be continuous, and then second, apply the well-known correspondence between sheaves and *local homeomorphisms* to obtain the desired map $\sigma : \mathsf{LB}_1T \to \mathsf{LB}_0T$.

In order to describe this sheaf, we need a way to talk about systems for generating sheaves by generators and relations. Luckily, our base space LB_0T being strongly zero-dimensional makes this quite a pleasant experience, because sheaves over LB_0T are "almost" expressible as algebraic objects admitting an action by the Grothendieck Boolean algebra $B_{\mathcal{J}}$ generating LB_0T. These algebraic objects are called $B_{\mathcal{J}}$-sets, whose theory we now introduce.

4.1 Sheaves, Local Homeomorphisms & $B_{\mathcal{J}}$-Sets

An important aspect of our results is that the source map $\sigma\colon \mathsf{LB}_1T \to \mathsf{LB}_0T$ of the localic behaviour category is a local homeomorphism. Here, a map f of locales is a *local homeomorphism* if there is a cover $\{ v_i \}_i$ of its domain such that, on each part of this cover, the map f restricts to an open injection. It is well-known that local homeomorphisms into a locale correspond to sheaves on a locale; and since LB_0T is ultraparacompact, this leads to a particularly appealing description of the source map. For indeed, sheaves on an ultraparacompact locale—or at least, those possessing a global section—can be described purely algebraically as sets with a suitable action of the corresponding Grothendieck Boolean algebra. This was first shown by Bergman [6] for Boolean algebras, and later extended to the Grothendieck case [11]. One advantage of this presentation is that it makes clear what homomorphisms and congruences of $B_{\mathcal{J}}$-sets are.

Definition 4.1 ($B_{\mathcal{J}}$-sets) Let B be a non-degenerate Boolean algebra (i.e., $0 \neq 1$ in B). A B-set F consists of a set $|F|$ equipped with one binary operation $b(-,-)$ for each $b \in B$ satisfying the equations

$$\begin{array}{lll} b(x,x) = x & b(b(x,y),z) = b(x,z) & b(x,b(y,z)) = b(x,z) \\ \top(x,y) = x & (\neg b)(x,y) = b(y,x) & (b \wedge c)(x,y) = b(c(x,y),y) \end{array} \tag{1}$$

for all $x, y, z \in |F|$. If $\mathcal{J}$ is a strongly zero-dimensional topology on B, then a $B_{\mathcal{J}}$-set F consists of a B-set F further equipped with a P-ary operation $P\colon |F|^P \to |F|$ for each partition $P \in \mathcal{J}$. These operations are required to satisfy, for any $z \in |F|$ and families $x, y \in |F|^P$, the axioms

$$P(\lambda b.z) = z \qquad P(\lambda b.b(x_b, y_b)) = P(\lambda b.x_b) \qquad \text{and} \qquad b(P(x), x_b) = x_b\ . \tag{2}$$

These axioms are rather intuitive if one reads each operation b as an if-then-else operation, and the infinitary operations P as infinitary switch statements. To see the correspondence with sheaves, view the elements of a $B_{\mathcal{J}}$-set as a global section. Then the operations perform amalgamation: for example $b(x,y)$ is the unique amalgam of $x|_b$ and $y|_{\neg b}$. We don't have to explicitly track local sections because if we have any global section t at all, then a local section s over b manifests as a global section by taking the amalgamation of s and $t|_{\neg b}$. Hence, the category of non-empty $B_{\mathcal{J}}$-sets is equivalent to the category of sheaves over the locale presented by $B_{\mathcal{J}}$ that have a global section.

Because every local section of a $B_{\mathcal{J}}$-set F comes from some global section, the set of local sections over some b is a quotient of $|F|$, by the relation $\equiv_b$ defined as follows.

Proposition 4.2 *[12, Proposition 2.6] Let $B_\mathcal{J}$ be a non-degenerate Grothendieck Boolean algebra. Any $B_\mathcal{J}$-set structure on a set $|F|$ induces equivalence relations $\equiv_b$ for $b \in B$ given by $x \equiv_b y \iff b(x,y) = y$. These equivalence relations satisfy:* (i) *if $x \equiv_b y$ and $c \leq b$ then $x \equiv_c y$;* (ii) *$x \equiv_\top y$ iff $x = y$, and $x \equiv_\bot y$ always;* (iii) *for any $P \in \mathcal{J}$ and $x \in X^P$, there is a unique $z \in X$ such that $z \equiv_b x_b$ for all $b \in P$. In fact, any B-indexed family of equivalence relations on $|F|$ satisfying (i)–(iii) determine a $B_\mathcal{J}$-set, wherein $P(\lambda b.x_b)$ is the aforementioned unique z. With this alternative definition, a $B_\mathcal{J}$-set homomorphism is a function that preserves the $\equiv_b$ relations.*

The sheaf corresponding to the source map will be constructed as a quotient of a free $B_\mathcal{J}$-set, so it is instructive to construct the free $B_\mathcal{J}$-set explicitly.

Definition 4.3 (Free $B_\mathcal{J}$-sets) Let A be a set and $B_\mathcal{J}$ a non-degenerate Grothendieck Boolean algebra. Then the *Grothendieck Boolean power* $A[B]^\mathcal{J}$ is the set of functions $h\colon A \to B$ for which $\{\, h(a) \mid a \in A \,\}^- \in \mathcal{J}$.

Remark 4.4 Notice that the following construction is precisely the monad of partitions T_B when taking $\mathcal{J}$ to be the topology of finite partitions. In other words, the B-sets are precisely the Eilenberg-Moore algebras of T_B, and definition 4.1 is the equational presentation of T_B. Of course, we can equally well define a monad of partitions $T_{B_\mathcal{J}}$ for Grothendieck Boolean algebras.

Proposition 4.5 *[11, Remark 3.17] Let A be a set. Then $A[B]^\mathcal{J}$ has a $B_\mathcal{J}$-set structure given by $P(\lambda b.h_b) := \lambda a. \bigvee_b b \wedge h_b(a)$, and this is the free $B_\mathcal{J}$-set with A-many generators. The unit map $A \to A[B]^\mathcal{J}$ identifies $a \in A$ with the map δ_a for which $\delta_a(a) := \top$ and $\delta_a(a') := \bot$ for $a' \neq a$.*

Given a sheaf F on a locale L, the corresponding local homeomorphism $E(F) \to L$ is found by taking $\mathcal{O}(E(F))$ as the frame of subsheaves of F. We can re-express this in terms of the category of sheaves on L: this is a topos, and in particular admits a subobject classifier Ω, so subsheaves of F correspond to sheaf maps $F \to \Omega$. Now if L is the ultraparacompact locale presented by $B_\mathcal{J}$, then Ω itself is a $B_\mathcal{J}$-set, and so the frame $\mathcal{O}(E(F))$ is given by the set of $B_\mathcal{J}$-set homomorphisms $F \to \Omega$ under pointwise ordering. Ω as a $B_\mathcal{J}$-set turns out to be $B_\mathcal{J}$ itself with the action $P(\lambda b.u_b) = \bigvee_{b \in P}(u_b \wedge b)$, or equivalently with $u \equiv_b v \iff b \wedge u = b \wedge v$.

Definition 4.6 Let L be an ultraparacompact locale presented by $B_\mathcal{J}$, and F be a $B_\mathcal{J}$-set. The *étale space* $E(F)$ corresponding to F is given by $\mathcal{O}(E(F)) := \mathsf{Set}_{B_\mathcal{J}}(F, \mathcal{O}L)$, and its associated *projection map* $\sigma\colon E(F) \to L$ is defined by $\sigma^{-1}\colon u \mapsto \mathsf{const}_u$ (the constant function at u).

Lemma 4.7 *Let L be an ultraparacompact locale presented by $B_\mathcal{J}$, and F be a $B_\mathcal{J}$-set. Each element $x \in |F|$ injectively induces an open $\hat{x} \in \mathcal{O}(E(F))$ defined by $\hat{x} := \lambda y. \bigvee \{\, b \mid x \equiv_b y \,\}$. Moreover, these generate $\mathcal{O}E(F)$ because every $w \in \mathcal{O}(E(F))$ can be expressed as $w = \bigvee_{x \in |F|} \hat{x} \wedge \mathsf{const}_{w(x)}$.*

Proposition 4.8 *Let L be an ultraparacompact locale presented by $B_\mathcal{J}$. Then the corresponding local homeomorphism of a $B_\mathcal{J}$-set F is the map $\sigma\colon E(F) \to L$.*

For a sheaf F, the points of the corresponding local homeomorphism $E(F)$ are known as *germs*. Here is the corresponding notion for $B_\mathcal{J}$-sets.

Proposition 4.9 *Let L be an ultraparacompact locale presented by $B_\mathcal{J}$ and F a $B_\mathcal{J}$-set. Then $\mathsf{pt}\, E(F) \cong \sum_{p \in \mathsf{pt}\, L} |F|/_{\equiv_p}$, where $x \equiv_p y \iff \exists b \ni p. x \equiv_b y$. An element $[x]_p$ of this space is called a* germ. *The topology on this space is generated by subbasic open sets $[x|b] = \{\, [x]_p \mid p \in b \,\}$.*

4.2 The Locale of Morphisms

Let $B_\mathcal{J}$ denote the Grothendieck Boolean algebra of complemented opens in $\mathcal{O}\mathsf{LB}_0T$. We can now construct the $B_\mathcal{J}$-set which will eventually become our locale of morphisms LB_1T. We know that this $B_\mathcal{J}$-set should be generated by computations in T, but it should not be generated *freely*: we can see in $\mathbb{B}T$ that if a term factors as $t \gg\!= u$, then at each $\beta \in [t \mapsto a]$ we have $t \gg\!= u \sim_\beta t \gg u(a)$, so the global sections $\eta(t \gg\!= u)$ and $\eta(t \gg u(a))$ are equal when restricted to the region $[t \mapsto a]$. Here the advantage of the $B_\mathcal{J}$-set approach becomes clear: the condition relating $t \gg\!= u$ with $t \gg u(a)$ above can be expressed purely equationally, as the following congruence relation on the $B_\mathcal{J}$-set freely generated by $T1$.

Definition 4.10 Let T be a monad. The *sheaf of transitions* F_T associated to T is a quotient of the free $B_\mathcal{J}$-set $T1[B]^\mathcal{J}$ with generators $T1$, by the smallest $B_\mathcal{J}$-congruence identifying $t \gg\!= u \approx P^{(t)}(\lambda[t \mapsto a].t \gg u(a))$ where $t \in TA$, $u\colon A \to T1$ and $P^{(t)} = \{\, [t \mapsto a] \mid a \in A \,\}^-$ is the partition canonically associated to T.

By our previous considerations, we expect the local homeomorphism constructed from F_T to be the locale of morphisms for our localic behaviour category. But it is not at all obvious what its relationship is with the morphisms of the behaviour category introduced in definition 2.8. To see the connection, we prove that two elements $x, y \in T1[B]^\mathcal{J}$ are related by $\approx$ just when they are pointwise trace-equivalent, as expressed by the following definition and accompanying lemma.

Definition 4.11 Let T be a monad of rank κ. Define the LB_0T-valued relation of *trace equivalence* on $T1$:

$$[\![m \sim m']\!] = \bigvee_{k\geq 1} [\![m \sim_k m']\!] \quad \text{where} \quad [\![m_1 \sim_k m_k]\!] = \bigvee \{\, \textstyle\bigwedge_{i=1}^{k-1} [\![m_i \sim_1 m_{i+1}]\!] \mid m_2 \dots m_{k-1} \in T1 \,\} \tag{3}$$

$$[\![m \sim_1 m']\!] = \bigvee \left\{ [t \mapsto a] \;\middle|\; \begin{array}{c} A \in \mathsf{Set}, |A| \leq \kappa, t : TA, u, u' \colon A \to T1, a \in A, \\ u(a) = u'(a), m = t \gg\!= u, m' = t \gg\!= u' \end{array} \right\} \tag{4}$$

If $u \in \mathcal{O}\mathsf{LB}_0T$ is a complemented open, we define $m \sim_u m'$ to be true whenever $u \leq [\![m \sim m']\!]$ (if $u = \top$, we simply write $m \sim m'$). This definition seems complicated, but it is just the point-free transliteration of the definition for β-equivalence. As β-equivalence is an equivalence relation, so is $[\![- \sim -]\!]$, as the following lemma shows.

Lemma 4.12 *$[\![- \sim -]\!]$ is an equivalence relation, in the sense that for all $m_1, m_2, m_3 \in T1$,*

$$[\![m_1 \sim m_1]\!] = \top \qquad [\![m_1 \sim m_2]\!] \leq [\![m_2 \sim m_1]\!] \qquad [\![m_1 \sim m_2]\!] \wedge [\![m_2 \sim m_3]\!] \leq [\![m_1 \sim m_3]\!].$$

Consequentially, all the $\sim_u$ are equivalence relations in the usual sense.

Now, we come to the central lemma relating $\approx$ to the more familiar β-equivalence. The proof of this lemma is quite involved, which we relegate to appendix A.9.

Lemma 4.13 *Let $x, y \in T1[B]^\mathcal{J}$. Then $x \approx y$ iff for each $m, n \in T1$, we have $m \sim_{x(m)\wedge y(n)} n$. Moreover, $m \equiv_b n \iff m \sim_b n$.*

Be warned that we abuse notation by confusing elements of $T1$ with the induced element of F_T, even though the unit map $\delta\colon T1 \to |F_T|$ from proposition 4.5 is in general not injective.

Recall from proposition 4.8 that the corresponding local homeomorphism is the locale of homomorphisms $\mathsf{Set}_{B_\mathcal{J}}(F_T, \mathsf{LB}_0T)$. By the universal property of free algebras, such homomorphisms correspond to functions $w\colon T1 \to \mathsf{LB}_0T$ respecting the generating equation $w(t \gg\!= u) = P^{(t)}(\lambda[t \mapsto a].t \gg u(a))$. We can restate this in terms of trace equivalence between generators, as follows (proof of correspondence can be found in appendix A.10).

Definition 4.14 The *locale of transitions* LB_1T is the pointwise-ordered poset of functions $w\colon T1 \to \mathcal{O}(\mathsf{LB}_0T)$ for which $m_1 \sim_b m_2$ implies $w(m_1) \equiv_b w(m_2)$ for any $m_1, m_2 \in T1$ and $b \in B$.

We are now in the position to introduce the localic behaviour category, but before we do so we specialize lemma 4.7 and proposition 4.9 to our locale of transitions, which relates the localic behaviour category back to the usual behaviour category.

Lemma 4.15 *Every open $w \in \mathcal{O}(\mathsf{LB}_1T)$ can be expressed as $w = \bigvee_m \hat{m} \wedge \mathsf{const}_{w(m)}$, so the frame $\mathcal{O}(\mathsf{LB}_1T)$ is generated by opens of the form $[m|b] := \lambda n.[\![m \sim n]\!] \wedge [b]$ for $m \in T1$ and $b \in T2$.*

Proposition 4.16 *The set of points $\mathsf{pt}(\mathsf{LB}_1T)$ is bijective with $\mathbb{B}_1T$.*

Finally, the following lemma is useful for we will often have to consider various pullbacks with the source map, such as when we define the composition map of the localic behaviour category in definition 2.8 below.

Lemma 4.17 *The pullback $L \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T$ of a locale map $f\colon L \to \mathsf{LB}_0 T$ along the source map $\sigma\colon \mathsf{LB}_1 T \to \mathsf{LB}_0 T$ has frame of opens given by the pointwise-ordered poset of functions $h\colon T1 \to \mathcal{O}L$ for which $m_1 \sim_b m_2$ implies $h(m_1) \wedge f^{-1}b = h(m_2) \wedge f^{-1}b$ for any $m_1, m_2 \in T1$ and $b \in B$.*

In terms of points, such a function h contains all the points $(x, [m]_\beta)$ for which $x \in h(m)$ and $f(x) = \beta$.

Definition 4.18 Let T be a monad. Then the *localic behaviour category* $\mathsf{LB}T$ has:

- locale of objects $\mathsf{LB}_0 T$ given by the terminal localic T-comodel;
- source map $\sigma\colon \mathsf{LB}_1 T \to \mathsf{LB}_0 T$ given by $\sigma^{-1}(u) := \mathsf{const}_u$;
- target map $\tau\colon \mathsf{LB}_1 T \to \mathsf{LB}_0 T$ given by $\tau^{-1}(u) := \lambda m.(\!|m|\!)^{-1}u$;
- identity map $\iota\colon \mathsf{LB}_0 T \to \mathsf{LB}_1 T$ given by $\iota^{-1}(w) := w(\mathsf{return})$;
- composition map $\mu\colon \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \to \mathsf{LB}_1 T$ given by $\mu^{-1}\colon w \mapsto \lambda m, n.w(m \gg n)$, where, by lemma 4.17 we identify $\mathcal{O}(\mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T)$ with the poset of functions $h\colon T1 \times T1 \to \mathcal{O}(\mathsf{LB}_0 T)$ for which $m_1 \sim_b m_2$ implies $h(m_1, n) \equiv_b h(m_2, n)$ and $n_1 \sim_b n_2$ implies $h(m, n_1) \equiv_{(\!|m|\!)^{-1}b} h(m, n_2)$.

A function $h\colon T1 \times T1 \to \mathcal{O}(\mathsf{LB}_0 T)$ as above corresponds to the open set containing pairs of germs $([m]_\beta, [n]_{\partial_m \beta})$ with $\beta \in h(m, n)$. For the verification that this is a localic category, see appendix A.13.

4.3 Topological Behaviour Category & Finitary Monads

We have already seen that $\mathsf{pt}\,\mathsf{LB}_0 T \cong \mathbb{B}_0 T$ (proposition 3.20) and $\mathsf{pt}\,\mathsf{LB}_1 T \cong \mathbb{B}_1 T$ (proposition 4.16). Being a right adjoint, the functor pt preserves limits and hence lifts to internal categories. Hence we get a *topological behaviour category* $\mathbb{B}T := \mathsf{pt}(\mathsf{LB}T)$, which is just an appropriately topologized version of the behaviour category.

Definition 4.19 (Topological behaviour category) Let T be a monad. The *operational topology* on $\mathbb{B}_1 T$ is generated by subbasic opens of the form $[m|b] := \{\, [m]_\beta \mid \beta \in [b] \,\}$. Taking $\mathbb{B}_0 T$ and $\mathbb{B}_1 T$ with their operational topologies makes the structure maps of the behaviour category continuous, yielding the *topological behaviour category* $\mathbb{B}T$.

We have also shown that for finitary monads T, $\mathsf{LB}_0 T$ is spatial and hence corresponds to $\mathbb{B}_0 T$. We ought to now establish also the spatiality of $\mathsf{LB}_1 T$, in order to conclude that it suffices to consider the topological behaviour category in the finitary monad case. Now, essentially $\mathsf{LB}_1 T$ is spatial because it is determined by a sheaf F_T over a spatial locale $\mathsf{LB}_0 T$, and sheaves only depend on the lattice of opens of its base space. More precisely, we observe in the following lemma that the source map $\sigma\colon \mathbb{B}_1 T \to \mathbb{B}_0 T$ of the topological behaviour category is also a local homeomorphism, and it has the same sections as the (source map of the) localic behaviour category, and is thus the same sheaf.

Lemma 4.20 *Let T be a finitary monad. Then for any global section s of $\sigma\colon \mathbb{B}_1 T \to \mathbb{B}_0 T$, there is a finite family of pairs $\{\, (b_i \in \mathsf{BB}_0 T, m_i \in T1) \,\}_{i \in I}$ such that $\{\, b_i \,\}_{i \in I}$ is a finite partition and s maps $\beta \in b_i \mapsto [m_i]_\beta$. Moreover, this family is unique up to trace equivalence: if we have two such families $\{\, (b_i, m_i) \,\}_{i \in I}$ and $\{\, (b_j, n_j) \,\}_{j \in J}$ then for all i, j we have $m_i \sim_{b_i \wedge b_j} n_j$.*

It is not hard to see that a family as in the lemma above defines an element of the free $\mathsf{BB}_0 T$-set, and that the uniqueness translates to the same condition as in lemma 4.13. That is, the global sections of $\mathbb{B}T$ correspond to the sheaf of transitions over $\mathsf{BB}_0 T$, and hence $\mathcal{O}(\mathbb{B}_1 T) \cong \mathsf{LB}_1 T$. Hence we have:

Proposition 4.21 *For finitary T, $\mathsf{LB}_0 T$ and $\mathsf{LB}_1 T$ are spatial, i.e., $\mathsf{LB}T$ has enough states and transitions.*

5 The Stone Adjunction for Monads

In this section, we functorialize the construction of the localic behaviour category and prove that it has a right adjoint by taking sections of the source map. Here, the correct morphisms between localic categories are *retrofunctors*, not functors like usual. Just as we view topological/localic categories as transition systems, we can view a retrofunctor $\mathbb{C} \rightsquigarrow \mathbb{D}$ as a *simulation* of transition systems.

Definition 5.1 [7, Example 2.9] Let $\mathbb{C}$ and $\mathbb{D}$ be small categories. A *retrofunctor* $F\colon \mathbb{C} \rightsquigarrow \mathbb{D}$ consists of two functions $F_0\colon \mathbb{C}_0 \to \mathbb{D}_0$ and $F_1\colon \mathbb{C}_0 \times_{\mathbb{D}_0} \mathbb{D}_1 \to \mathbb{C}_1$, where $\mathbb{C}_0 \times_{\mathbb{D}_0} \mathbb{D}_1$ is the pullback of F_0 along $\sigma_{\mathbb{D}}$. In other words, given $c \in \mathbb{C}_0$ and $f \in \mathbb{D}(F_0 c, d)$ we get a *lift* $F_1(c,f)\colon c \to c'$ such that $F_0 c' = d$. These are further required to respect identity and composition:

$$F_1(c, \mathsf{id}_{F_0 c}) = \mathsf{id}_c \qquad F_1(c, g \circ f) = F_1(c', g) \circ F_1(c, f) \quad \text{where} \quad F_1(c,f)\colon c \to c' \tag{5}$$

If $\mathbb{C}$ and $\mathbb{D}$ are internal categories, then there is an appropriate notion of internal retrofunctor which make the appropriate diagrams commute, as described by Clarke [7, definition 2.10]. Write $\mathsf{LocRetro}$ and $\mathsf{TopRetro}$ for the categories of internal categories and retrofunctors in Loc and Top respectively.

Proposition 5.2 *The assignment $T \mapsto \mathsf{LB}T$ extends contravariantly to a functor $\mathsf{LB}\colon \mathsf{Mnd}_r(\mathsf{Set}) \to \mathsf{LocRetro}^{\mathsf{op}}$, and similarly a functor $\mathbb{B}\colon \mathsf{Mnd}_\omega(\mathsf{Set}) \to \mathsf{TopRetro}^{\mathsf{op}}$.*

Proof. A monad morphism $\varphi\colon T \to S$ induces a retrofunctor whose action on objects $\mathsf{LB}_0\varphi\colon \mathsf{LB}_0 S \to \mathsf{LB}_0 T$ is given on generating opens by $(\mathsf{LB}_0\varphi)^{-1}\colon [b] \mapsto [\varphi(b)]$. For the action on morphisms $(\mathsf{LB}_1\varphi)\colon \mathsf{LB}_0 S \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \to \mathsf{LB}_1 S$, by lemma 4.17 we can identify $\mathcal{O}(\mathsf{LB}_0 S \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T)$ with an appropriate poset of functions $h\colon T1 \to \mathcal{O}(\mathsf{LB}_0 S)$. The action is then simply given by $(\mathsf{LB}_1\varphi)^{-1}\colon w \mapsto w \circ \varphi_1$. We refer to appendix A.15 for more details and the verification of functoriality. □

On the other hand, if we view a localic category LC as a transition system, what is a computation on LC? Well, a computation (of output type A) should specify, for each state $c \in \mathsf{LC}_0$, a transition out of c (the "side-effect" of the computation) along with an output in A. In other words, the computations are *global sections* of the source map, or more precisely a disjoint, A-indexed, jointly global family of partial sections. Indeed, we get a monad of such sections—notice that this looks very much like a state monad except we also keep track which transitions are taken, not just the end state.

Proposition 5.3 *Let LC be a localic category. Then the endofunctor*

$$\Gamma\mathsf{LC}(A) = \{\, s\colon \mathsf{LC}_0 \to A \cdot \mathsf{LC}_1 \mid \mathsf{id}_{\mathsf{LC}_0} = \mathsf{LC}_0 \xrightarrow{s} A \cdot \mathsf{LC}_1 \xrightarrow{\pi_{\mathsf{LC}_1}} \mathsf{LC}_1 \xrightarrow{\sigma} \mathsf{LC}_0 \,\}$$

on Set (with the action on $f\colon A \to B$ given by post-composing $f \cdot \mathsf{LC}_1$) admits a monad structure given, for arbitrary $a \in A$, $s \in \Gamma\mathsf{LC}A, u\colon A \to \Gamma\mathsf{LC}B$ by $\mathsf{return}\, a = \mathsf{LC}_0 \xrightarrow{\iota} \mathsf{LC}_1 \xrightarrow{v_a} A \cdot \mathsf{LC}_1$ and

$$s \gg\!= u = \mathsf{LC}_0 \xrightarrow{s} A \cdot \mathsf{LC}_1 \xrightarrow{\langle \pi, u \circ (A \cdot \tau)\rangle} \mathsf{LC}_1 \times (B \cdot \mathsf{LC}_1) \xrightarrow{\cong} B \cdot (\mathsf{LC}_1 \times \mathsf{LC}_1) \xrightarrow{\mu} B \cdot \mathsf{LC}_1.$$

In terms of points, we have $\mathsf{return}\, a\colon c \mapsto (a, \mathsf{id}_c)$ and $s \gg\!= u\colon c \mapsto (b, g \circ f)$ where $(a,f) =: s(c)$ and $(b,g) =: u(a)(\tau(f))$, while in terms of opens we have

$$\begin{aligned}
(\mathsf{return}\, a)^{-1} &: \langle a' \mapsto w\rangle \mapsto \text{if } a = a' \text{ then } \iota^{-1} w \text{ else } \bot \\
(s \gg\!= u)^{-1} &: \langle b \mapsto w\rangle \mapsto \bigvee_{a \in A} s^{-1} \Big\langle a \mapsto \bigvee \{\, v_1 \wedge \tau^{-1} u(a)^{-1} \langle b \mapsto v_2\rangle \mid v_1 \times v_2 \leq \mu^{-1}(w) \,\} \Big\rangle .
\end{aligned}$$

Moreover, the assignment $\mathsf{LC} \mapsto \Gamma\mathsf{LC}$ defines a contravariant functor $\Gamma\colon \mathsf{LocRetro}^{\mathsf{op}} \to \mathsf{Mnd}_r(\mathsf{Set})$.

Proof sketch. A straightforward diagram chase reveals that unitality and associativity of the monad structure is inherited from the unitality and associativity of LC. It is a basic theorem of category theory that the functorial action of a right adjoint functor is determined by the adjunction, so functoriality is automatically obtained when we prove the adjunction of theorem 5.4. Since we don't use the functorial action of Γ, we leave it as an exercise to define the action. □

Any computation $t \in TA$ defines a global section $\eta(t)\colon \mathsf{LB}_0 T \to A \cdot \mathsf{LB}_1 T$ of the behaviour category, defined by $\eta(t)^{-1}\colon \langle a \mapsto w\rangle \mapsto [t \mapsto a] \wedge w(t \gg \mathsf{return})$ on generating opens. This defines the unit map of an adjunction between LB and Γ, which brings us to the main adjunction of this paper.

Theorem 5.4 $\mathsf{LB} : \mathsf{Mnd}_r(\mathsf{Set}) \underset{\Gamma}{\overset{\mathsf{LB}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{LocRetro}^{\mathsf{op}} : \Gamma.$

Proof sketch. The counit map $\varepsilon\colon \mathsf{LC} \rightsquigarrow \mathsf{LB\Gamma LC}$ is given by $\varepsilon_0^{-1}\colon [s] \mapsto s^{-1}\,\langle 1 \mapsto \top\rangle$, and $\varepsilon_1^{-1}\colon u \mapsto \lambda m \in T1.m^{-1}u$. See appendix A.16 for the verification of the adjunction. □

We also have a functor $\mathbb{B} := \mathsf{pt\,LB}\colon \mathsf{Mnd}_r(\mathsf{Set}) \to \mathsf{TopRetro}^{\mathsf{op}}$, and this similarly admits a right adjoint Γ, but $\mathbb{B}T$ loses too much information about the infinitary monad T, as exemplified by example 3.15. However, by proposition 4.21, if we restrict to $T \in \mathsf{Mnd}_\omega(\mathsf{Set})$ then $\mathsf{LB}T$ is spatial and corresponds to the topological category $\mathbb{B}T$. The right adjoint of the functor $\mathbb{B}\colon \mathsf{Mnd}_\omega(\mathsf{Set}) \to \mathsf{TopRetro}^{\mathsf{op}}$ is given by taking the monad of *finitary sections* $\Gamma_\omega\mathbb{C}$ for a topological category $\mathbb{C}$.

Theorem 5.5 $\mathbb{B} : \mathsf{Mnd}_\omega(\mathsf{Set}) \underset{\longleftarrow}{\overset{\longrightarrow}{\bot}} \mathsf{TopRetro}^{\mathsf{op}} : \Gamma_\omega$.

Proof. The inclusion $i\colon \mathsf{Mnd}_\omega(\mathsf{Set}) \hookrightarrow \mathsf{Mnd}_r(\mathsf{Set})$ has a right adjoint given by $\mathsf{lan}_j(- \circ j)$ where $j\colon \mathsf{Set}_\omega \hookrightarrow \mathsf{Set}$ is the usual full inclusion of the category of finite sets (as follows from relative monad theory [5]). Then we have the desired adjunction by composing $\mathbb{B} \dashv \Gamma$ with $i \dashv \mathsf{lan}_j(- \circ j)$, noting that $\Gamma_\omega := \mathsf{lan}_j(\Gamma(-) \circ j)$.□

6 The Stone Duality for Hyperaffine-Unary Monads

This final section is devoted to proving that adjunctions 5.4 and 5.5 are idempotent, and to characterize their fixed points. On the monad side, the fixed points correspond to those monads with a Cartesian-closed category of Eilenberg-Moore algebras. These were first syntactically characterized by Johnstone [17], but the first-named author provided an improved characterization [11] as those monads which admit a *hyperaffine-unary decomposition*. On the side of localic categories, the fixed points are the *ample localic categories*, i.e., whose source maps are local homeomorphisms and whose locales of objects are ultraparacompact.

In fact, this equivalence between hyperaffine-unary monads and ample localic categories is originally due to the first-named author [12]. In addition to filling in details about the equivalence, our contribution is to envelope the equivalence in an adjunction, which yields a process of *hyperaffine-unary completion* for monads on one hand, and a process of *amplification* for localic categories on the other.

6.1 Hyperaffine-Unary Monads

What exactly is the difference between T and $\Gamma\mathsf{LB}T$? Because of the unit map, all computations in T live inside $\Gamma\mathsf{LB}T$. The answer is that T adds additional operations $\bar{t}$ which predict the output of t without performing the side effect of t. Computationally, this can be thought of as performing t and then rolling the state back, which we will refer to as *scrying*. In the case where T encodes computations consuming from a stream [4], the newly added scrying operations look ahead into the stream without consuming from it, as the following example demonstrates.

Example 6.1 (Binary Input) Let T be the monad of binary input, whose terminal topological comodel $\mathbb{B}_0T$ is the cantor space of example 3.4. By continuity and compactness of $\mathbb{B}_0T$, any global section $s \in \Gamma\mathbb{B}T(A)$ is therefore described (non-uniquely) by a pair $(B, |s|\colon B \to \mathbb{N} \times A)$ where B is a finite set of finite strings $B \subseteq 2^{<\omega}$ that jointly cover all infinite streams, and $|s|$ assigns to each element of B a pair (n, a) consisting of the number n of digits to consume from the stream, and the output a. As an example, the binary tree $t = b(a_0, b(b(a_1, a_2), a_3)) \in TA$, induces a section $\eta(t)$ which is described by the assignments $\{\, 0 \mapsto (1, a_0); 100 \mapsto (3, a_1); 101 \mapsto (3, a_2); 11 \mapsto (2, a_3)\,\}$.

In general, an assignment $\mathfrak{b} \mapsto (n, a)$ for a section of the form $\eta(t)$ must satisfy $n \geq \mathsf{length}(\mathfrak{b})$. That is, to use information about the first $k = \mathsf{length}(\mathfrak{b})$ digits of the stream you must consume at least k digits. However, in general sections do not need to respect this: the assignments $\{\, 0 \mapsto (0, a_0); 10 \mapsto (1, a_1); 11 \mapsto (1, a_2)\,\}$ describe a perfectly acceptable section s. We can think of s as *looking ahead at* or *scrying* the first two digits of the stream, before deciding what to do. Indeed, for any section s, we have a corresponding $\overline{s} \in \Gamma\mathbb{B}T(A)$ which outputs the same values as s, but only makes identity transitions. Then s factors as $s = \overline{s} \gg\!= \lambda a.s \gg \mathsf{return}\, a$.

[4] A particularly potent analogy is to think of the environment as an (infinite) deck of playing cards, and of the program as the player, in which case a scry allows the player to look at the top $n \in \mathbb{N}$ cards of their deck before putting it back in the same order. Players of a certain popular trading card game will recognize this, but note that "scrying" there allows you to reorder the cards (and to put them at the bottom of the deck).

It is easy to see in general that monads of the form $\Gamma\mathsf{LC}$ always have this factorization property, since a section s always admits a cousin $\overline{s}$ which sends objects to the same output, but replaces the morphism by identity morphisms—the *scry* corresponding to s. Monads satisfying this factorization property are called *hyperaffine-unary theories* in [11], and they suffice to characterize the fixed points of adjunction 5.4.

Definition 6.2 (Hyperaffine-unary) Let T be a monad. A computation $h \in TA$ is *hyperaffine* if

$$h \gg \mathsf{return}\, a = \mathsf{return}\, a \qquad \text{and} \qquad h \gg\!= \lambda a_1.h \gg\!= \lambda a_2.\, \mathsf{return}(a_1, a_2) = h \gg\!= \lambda a.\, \mathsf{return}(a, a). \quad (6)$$

The monad T is *hyperaffine-unary* if for every computation $t \in TA$, there is a unique hyperaffine $\overline{t} \in TA$ such that $t = \overline{t} \gg\!= \lambda a.(t \gg \mathsf{return}\, a)$. Any hyperaffine-unary monad admits a submonad H of hyperaffine operations ([11, Proposition 6.1]).

Proposition 6.3 *Let* LC *be a localic category. Then* $\Gamma\mathsf{LC}$ *is hyperaffine-unary.*

Hyperaffine-unary monads admit a particularly nice presentation of the localic behaviour category, which greatly aids us in proving the characterization of the fixed points.

Lemma 6.4 *Let T be a hyperaffine-unary monad with submonad of hyperaffines H. Then $\mathcal{O}(\mathsf{LB}_0 T)$ is generated by $H2_{\mathcal{J}}$ where $H2$ admits a Boolean algebra structure and $\mathcal{J}$ is a strongly zero-dimensional topology defined by $\{\, P^{(h)} \mid h \in HA \,\}$ with $P^{(h)} = \{\, [h \mapsto a] \mid a \in A \,\}$. Here, we abuse notation by writing $[h \mapsto a]$ for $h \gg\!= \lambda a'.\delta_a(a') \in H2$. Moreover, the map $\delta\colon T1 \to F_T$ is an isomorphism, with $T1$ admitting a $H2_{\mathcal{J}}$-set structure whose $P^{(h)}$-ary operations are defined by $P^{(h)}(\lambda a.m_a) := h \gg\!= \lambda a.m_a$.*

Proposition 6.5 *A monad T is hyperaffine-unary iff the unit map $\eta_T\colon T \to \Gamma\mathsf{LB}T$ of adjunction 5.4 is an isomorphism.*

6.2 Ample Localic Categories and Stone Duality

On the other side of the adjunction, what is the relationship between a localic category LC and the behaviour category $\mathsf{LB}\Gamma\mathsf{LC}$? Well, Γ only considers the partitioning sections of LC, so is only sensitive to the *ultraparacompact quotient* of LC_0, i.e., whose frame of opens is the ultraparacompact frame generated by taking the zero-dimensional topology of partitions on the Boolean algebra $\mathfrak{B}(\mathsf{LC}_0)$. Moreover, LB then reconstructs the locale of morphisms from only the sections over this ultraparacompact quotient. So this reconstruction is perfect if in the first place LC_0 is ultraparacompact and the source map is a local homeomorphism. These are called *ample localic categories* [12].

Definition 6.6 A localic category LC is *ample* if σ_{LC} is a local homeomorphism and LC_0 is ultraparacompact. A topological category $\mathbb{C}$ is *ample* if $\sigma_{\mathbb{C}}$ is a local homeomorphism and $\mathbb{C}_0$ is a Stone space. Write $\mathsf{AmpLocRetro}$ (resp. $\mathsf{AmpTopRetro}$) for the full subcategory of $\mathsf{LocRetro}$ (resp. $\mathsf{TopRetro}$) containing the ample localic (resp. topological) categories.

Proposition 6.7 *A localic category* LC *is ample iff the counit map* $\varepsilon_{\mathsf{LC}}\colon \mathsf{LC} \rightsquigarrow \mathsf{LB}\Gamma\mathsf{LC}$ *is an isomorphism.*

The combination of propositions 6.5 and 6.7 allow us to derive the titular Stone duality for monads.

Theorem 6.8 *The adjunction of theorem 5.4 is idempotent and its fixed points are the equivalent categories* $\mathsf{HUMnd}_{\mathsf{r}} \simeq \mathsf{AmpLocRetro}$. *Furthermore, this equivalence restricts to* $\mathsf{HUMnd}_{\omega} \simeq \mathsf{AmpTopRetro}$.

Example 6.9 Every Grothendieck Boolean algebra $B_{\mathcal{J}}$ presenting a locale L is associated to a distributions monad $T_{B_{\mathcal{J}}}(A) := A[B]^{\mathcal{J}}$ (4.4) whose computations are all hyperaffine, and hence hyperaffine-unary. Under the equivalence of theorem 6.8, $T_{B_{\mathcal{J}}}$ corresponds to the localic category $\mathsf{LB}T_{B_{\mathcal{J}}}$ with $\mathsf{LB}_0 T_{B_{\mathcal{J}}} \cong \mathsf{LB}_1 T_{B_{\mathcal{J}}} \cong L$ and source map $\sigma = \mathsf{id}\colon L \to L$. Of course, if $\mathcal{J}$ consists of only finite partitions, then $L = \mathsf{Spec}(B)$ is the Stone dual of B, and so our equivalence subsumes the classical Stone duality.

References

[1] Aguiar, M., *Internal Categories and Quantum Groups*, Cornell University (1997).

[2] Ahman, D. and A. Bauer, *Runners in Action*, pages 29–55, Lecture notes in computer science, Springer International Publishing (2020).
https://doi.org/10.1007/978-3-030-44914-8_2

[3] Ahman, D. and T. Uustalu, *Directed containers as categories*, Electronic Proceedings in Theoretical Computer Science **207**, pages 89–98 (2016).
https://doi.org/10.4204/eptcs.207.5

[4] Ahman, D. and T. Uustalu, *Taking updates seriously*, in: R. Eramo and M. Johnson, editors, *Proceedings of the Sixth International Workshop on Bidirectional Transformations*, pages 59–73, CEUR Workshop Proceedings (2017).

[5] Altenkirch, T., J. Chapman and T. Uustalu, *Monads need not be endofunctors*, in: C. L. Ong, editor, *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6014 of *Lecture Notes in Computer Science*, pages 297–311, Springer (2010).
https://doi.org/10.1007/978-3-642-12032-9_21

[6] Bergman, G. M., *Actions of boolean rings on sets*, Algebra universalis **28**, pages 153–187 (1991).
https://doi.org/10.1007/bf01190851

[7] Clarke, B., *Internal split opfibrations and cofunctors*, Theory and Applications of Categories **35**, pages 1608–1633 (2020).

[8] Cockett, R. and R. Garner, *Generalising the étale groupoid–complete pseudogroup correspondence*, Advances in mathematics **392**, page 108030 (2021).
https://doi.org/10.1016/j.aim.2021.108030

[9] Garner, R., *The costructure–cosemantics adjunction for comodels for computational effects*, Mathematical structures in computer science **32**, pages 374–419 (2022).
https://doi.org/10.1017/s0960129521000219

[10] Garner, R., *Stream processors and comodels*, Logical Methods in Computer Science **19, Issue 1** (2023).
https://doi.org/10.46298/lmcs-19(1:2)2023

[11] Garner, R., *Cartesian closed varieties I: the classification theorem*, Algebra universalis **85** (2024).
https://doi.org/10.1007/s00012-024-00869-1

[12] Garner, R., *Cartesian closed varieties II: links to algebra and self-similarity*, Proceedings of the Royal Society of Edinburgh: Section A Mathematics pages 1–45 (2025).
https://doi.org/10.1017/prm.2024.80

[13] Goncharov, S. and L. Schröder, *A relatively complete generic hoare logic for order-enriched effects*, in: *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, IEEE (2013).
https://doi.org/10.1109/lics.2013.33

[14] Harel, D., D. Kozen and J. Tiuryn, *Dynamic logic*, pages 99–217, Springer Netherlands (2001).
https://doi.org/10.1007/978-94-017-0456-4_2

[15] Johnstone, P., *Cartesian monads on toposes*, Journal of pure and applied algebra **116**, pages 199–220 (1997).
https://doi.org/10.1016/s0022-4049(96)00165-x

[16] Johnstone, P. T., *Stone Spaces*, Cambridge University Press (1982).

[17] Johnstone, P. T., *Collapsed toposes and cartesian closed varieties*, Journal of algebra **129**, pages 446–480 (1990).
https://doi.org/10.1016/0021-8693(90)90230-l

[18] Johnstone, P. T., *Sketches of an Elephant: Volume 2*, Oxford University Press (2002).

[19] Katsumata, S.-Y., E. Rivas and T. Uustalu, *Interaction laws of monads and comonads*, in: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, ACM (2020).
https://doi.org/10.1145/3373718.3394808

[20] Kelly, G. M. and A. J. Power, *Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads*, Journal of Pure and Applied Algebra **89**, pages 163–179 (1993).
https://doi.org/10.1016/0022-4049(93)90092-8

[21] Moggi, E., *Notions of computation and monads*, Information and Computation **93**, pages 55–92 (1991).
https://doi.org/10.1016/0890-5401(91)90052-4

[22] Niu, N. and D. I. Spivak, *Polynomial Functors: A Mathematical Theory of Interaction*, London Mathematical Society Lecture Note Series, Cambridge University Press (2025).

[23] Paterson, A. L. T., *Groupoids, inverse semigroups, and their operator algebras*, volume 170 of *Progress in Mathematics*, Birkhäuser (1999).

[24] Plotkin, G. and J. Power, *Notions of Computation Determine Monads*, pages 342–356, Lecture notes in computer science, Springer Berlin Heidelberg (2002).
https://doi.org/10.1007/3-540-45931-6_24

[25] Plotkin, G. and J. Power, *Computational effects and operations: An overview*, Electronic Notes in Theoretical Computer Science **73**, pages 149–163 (2004).
https://doi.org/10.1016/j.entcs.2004.08.008

[26] Power, J. and O. Shkaravska, *From comodels to coalgebras: State and arrays*, Electronic notes in theoretical computer science **106**, pages 297–314 (2004).
https://doi.org/10.1016/j.entcs.2004.02.041

[27] Uustalu, T., *Stateful runners of effectful computations*, Electronic notes in theoretical computer science **319**, pages 403–421 (2015).
https://doi.org/10.1016/j.entcs.2015.12.024

[28] Uustalu, T. and N. Voorneveld, *Algebraic and Coalgebraic Perspectives on Interaction Laws*, pages 186–205, Lecture notes in computer science, Springer International Publishing (2020).
https://doi.org/10.1007/978-3-030-64437-6_10

[29] Van Name, J., *Ultraparacompactness and ultranormality*, arXiv:1306.6086v1 [math.GN] (2013).

[30] Vickers, S., *Topology via Logic*, Cambridge University Press (1996).

A Omitted Proofs

A.1 Proof of proposition 2.13

This admits a rather lengthy "elementary" proof, but we can also prove it from other results in this paper, namely via the following chain of isomorphisms:

$$\mathbb{B}_0 T_B \overset{\text{prop } 3.11}{\cong} \mathsf{pt}\,\mathsf{LB}_0 T = \mathsf{Loc}(1, \mathsf{LB}_0 T_B) \overset{\text{lem. } 6.4}{\cong} \mathsf{Frm}(\mathsf{Idl}(B), S) \cong \mathsf{BA}(B, S) \cong \mathsf{Spec}(\mathsf{B})$$

where $S = \mathcal{O}1$ is the initial frame $\{ \bot \leq \top \}$, the penultimate isomorphism follows by universal property of the ideal construction as the left adjoint to the inclusion $BA \to \mathsf{Frm}$, and the last isomorphism identifies a map $p : B \to S$ with the ultrafilter $\mathfrak{p} = \{ b \mid p(b) = \top \}$.

A.2 Proof of proposition 3.14

We first check that $(\!|-|\!)$ respects $\gg\!=$ and return, making $\mathsf{LB}_0 T$ a comodel. For $\gg\!=$ we have

$$\begin{aligned}
&(\!|t \gg\!= u|\!)^{-1} \langle b_0 \mapsto [t_0] \rangle \\
=&[t \gg\!= u \mapsto b_0] \wedge [t \gg\!= u \gg t_0] && \text{by definition of } (\!|t \gg\!= u|\!) \\
=&\bigvee_{a \in A} [t \mapsto a] \wedge [t \gg u(a) \mapsto b_0] \wedge [t \gg u(a) \gg t_0] && \text{apply } (\mathsf{LB}_0\text{-}\mu) \text{ twice, and simplify} \\
=&\bigvee_{a} ([t \mapsto a] \wedge [t \gg u(a) \mapsto b_0]) \wedge ([t \mapsto a] \wedge [t \gg u(a) \gg t_0]) \\
=&\bigvee_{a} (\!|t|\!)^{-1} \langle a \mapsto [u(a) \mapsto b_0] \rangle \wedge (\!|t|\!)^{-1} \langle a \mapsto [u(a) \gg t_0] \rangle && \text{by definition of } (\!|t|\!) \\
=&(\!|t|\!)^{-1} \bigvee_{a} \langle a \mapsto [u(a) \mapsto b_0] \wedge [u(a) \gg t_0] \rangle && \text{by definition of } (\!|u|\!) \\
=&(\!|t|\!)^{-1} (\!|u|\!)^{-1} \langle b_0 \mapsto [t_0] \rangle ,
\end{aligned}$$

whereas for return, we compute

$$(\!|\mathsf{return}\ a|\!) \langle a_0 \mapsto [t_0] \rangle = [\mathsf{return}\ a \mapsto a_0] \wedge [\mathsf{return}\ a \gg t_0] = \begin{cases} [t_0] & \text{if } a = a_0 \\ \bot & \text{otherwise,} \end{cases}$$

but this is just v_a. Next, we show that this comodel is terminal, so let L be an arbitrary localic comodel. If a map $h \colon L \to \mathsf{LB}_0 T$ exists, then h being a comodel map implies $h^{-1}[t_0] = h^{-1}(\!|t_0|\!)^{-1} \langle 1 \mapsto \top \rangle = (\!|t_0|\!)_L^{-1} (2 \cdot h)^{-1} \langle 1 \mapsto \top \rangle = (\!|t_0|\!)_L^{-1} \langle 1 \mapsto \top \rangle$, and hence this uniquely determines h. We leave the verification that this map is well-defined as an exercise to the reader.

A.3 Proof of correspondence for example 3.15

Definition A.1 The locale of injective functions $\mathbb{R} \rightarrowtail \mathbb{N}$ is presented by generators $\langle x \mapsto n \rangle$ for $x \in \mathbb{R}$ and $n \in \mathbb{N}$, required to satisfy, for $x \neq y$ and $m \neq n$, the equations

$$\bigvee_{x \in \mathbb{R}} \langle x \mapsto n \rangle = \top \qquad \langle x \mapsto n \rangle \wedge \langle x \mapsto m \rangle = \bot \qquad \langle x \mapsto n \rangle \wedge \langle y \mapsto n \rangle = \bot$$

We must prove that this presentation is bi-interpretable with the behaviour locale of injective state. In one direction, the generator $\langle x \mapsto n \rangle$ is interpreted as $[\mathsf{get}_x \mapsto n]$, in which case the first two axioms

straightforwardly follow. The third axiom can be proven as follows:

$$\begin{aligned}
&[\mathsf{get}_x \mapsto n] \wedge [\mathsf{get}_y \mapsto n] \\
&= [\mathsf{get}_x \mapsto n] \wedge [\mathsf{get}_x \gg \mathsf{get}_y \mapsto n] \\
&= [\mathsf{get}_x \gg\!= \lambda m_1.\mathsf{get}_y \gg\!= \lambda m_2.\, \mathsf{return}(m_1, m_2) \mapsto (n, n)] \\
&= [\mathsf{get}_x \gg\!= \lambda m_1.\mathsf{get}_y \gg\!= \lambda m_2.\, \mathsf{return}(m_1 \stackrel{?}{=} n \stackrel{?}{=} m_2)] \\
&= [\mathsf{get}_x \gg\!= \lambda m_1.\mathsf{get}_y \gg\!= \lambda m_2.\, \mathsf{return}\, 0] && \text{(by injectivity eqn)} \\
&= [\mathsf{return}\, 0] = \bot
\end{aligned}$$

In the other direction, by recursion we define a map h interpreting the generators of the behaviour locale:

$$h\colon [\mathsf{return}\, 0] \mapsto \bot \qquad [\mathsf{return}\, 1] \mapsto \top \qquad [\mathsf{get}_x \gg\!= \lambda n.t_n] \mapsto \bigvee_{n\in\mathbb{N}} \langle x \mapsto n\rangle \wedge h([t_n])$$

We leave the reader to verify that this respects the usual equations satisfied by terms in the theory of state. As an example, we will verify just the injectivity equation from example 3.15:

$$\begin{aligned}
&h[\mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_y \gg\!= \lambda m.\, \mathsf{return}\, f(n,m) \mapsto (n_0, m_0)] \\
&= \bigvee_n \bigvee_m \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge f(n,m) \stackrel{?}{=} (n_0, m_0) \\
&= \left(\bigvee_{n\neq m} \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0)\right) \vee \bigvee_n \langle x \mapsto n\rangle \wedge \langle y \mapsto n\rangle \wedge f(n,m) \stackrel{?}{=} (n_0, m_0) && \text{property of } f \\
&= \left(\bigvee_{n\neq m} \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0)\right) \vee \bot && \text{by def. A.1} \\
&= \left(\bigvee_{n\neq m} \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0)\right) \vee \bigvee_n \bot \wedge (n,m) \stackrel{?}{=} (n_0, m_0) \\
&= \left(\bigvee_{n\neq m} \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0)\right) \vee \bigvee_n \langle x \mapsto n\rangle \wedge \langle y \mapsto n\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0) \\
&= \bigvee_n \bigvee_m \langle x \mapsto n\rangle \wedge \langle y \mapsto m\rangle \wedge (n,m) \stackrel{?}{=} (n_0, m_0) \\
&= h[\mathsf{get}_x \gg\!= \lambda n.\mathsf{get}_y \gg\!= \lambda m.\, \mathsf{return}(n,m) \mapsto (n_0, m_0)]
\end{aligned}$$

Finally, h respects the equations of the behaviour locale: for all three axioms (LB_0-$\bot$), (LB_0-η), and (LB_0-μ) it follows by a straightforward induction on the syntax of t.

A.4 Proof of proposition 3.19

We follow a similar line of argument to [15, Section 2]. First, notice that we can construct $\mathsf{LB}_0 T$ in two steps. Begin by constructing the meet semi-lattice $\mathsf{MB}_0\mathsf{T}$ generated by opens $[b]$ subject to equations $[t \gg \mathsf{return}\, a \mapsto a] = \top$ and $[t \mapsto a] \wedge [t \gg\!= u \mapsto b] = [t \mapsto a] \wedge [t \gg u(a) \mapsto b]$. Then we can generate the frame $\mathcal{O}\mathsf{LB}_0 T$ from $\mathsf{MB}_0 T$ subject to the equations $[t \mapsto a] \wedge [t \mapsto a'] = \bot$ and $\bigvee_{a\in A}[t \mapsto a] = \top$. Following [16, II 2.11], this can be presented as a covering system instead. The covering system $\mathcal{J}$ is generated by the

following rules:

$$\frac{t \in TA, a \neq a' \in A}{\emptyset \in \mathcal{J}([t \mapsto a] \wedge [t \mapsto a'])} \qquad \frac{t \in TA}{\{\, [t \mapsto a] \mid a \in A \,\} \in \mathcal{J}(\top)} \qquad \frac{}{\{\, u \,\} \in \mathcal{J}(u)}$$

$$\frac{J \in \mathcal{J}(u)}{\{\, j \wedge v \mid j \in J \,\} \in \mathcal{J}(u \wedge v)} \qquad \frac{J \in \mathcal{J}(u) \quad K_j \in \mathcal{J}(j) \;\; \forall j \in J}{\bigcup_{j \in J} K_j \in \mathcal{J}(u)}$$

Notice that the three base cases are pairwise-disjoint covers, and the inductive cases preserve the pairwise-disjoint property. Hence all the covers in this system are pairwise-disjoint. The frame presented by this system consists of all the $\mathcal{J}$-ideals, i.e., downwards closed subsets $I \subseteq \mathsf{MB}_0T$ such that for any $J \in \mathcal{J}(u)$, $J \subseteq I$ implies $u \in I$. For any subset $S \subseteq \mathsf{MB}_0T$, the smallest $\mathcal{J}$-ideal containing S is $\overline{S} = \{\, u \in \mathsf{MB}_0T \mid \exists J \in \mathcal{J}(u).J \subseteq S \,\}$, and the join of a family of $\mathcal{J}$-ideals $\{\, I_k \,\}_k$ is the $\overline{\bigcup_k I_k}$. Also, every $u \in \mathsf{MB}_0T$ induces an ideal $\downarrow u$.

We now prove that every open cover is refined by a partition, so consider an open cover $\{\, I_k \,\}_k$. It is covering iff $\top \in \overline{\bigcup_k I_k}$ iff there is $J \in \mathcal{J}(\top)$ such that $J \subseteq \bigcup_k I_k$. But if we now consider the family $\{\, \downarrow j \mid j \in J \,\}$, then this is precisely a partition (pairwise-disjoint because J is) refining $\{\, I_k \,\}_k$.

A.5 Proof of proposition 3.20

Suppose L is compact and ultraparacompact. This ensures that L is freely generated by the Boolean algebra $\mathfrak{B}(L)$, and locales freely generated from a distributive lattice in this way are well-known to be spatial [16, II 3.4]. $\mathsf{pt}\,L$ inherits zero-dimensionality and compactness from L, and it is Hausdorff because for any two points $\mathfrak{p} \neq \mathfrak{q} \in \mathsf{pt}\,L$, there must be a $b \in \mathfrak{B}(L)$ such that $\mathfrak{p} \in [b]$ and $\mathfrak{q} \in [\neg b]$, thus separating the two points.

On the other hand, if $L = \mathcal{O}X$ for a Stone space X, then the compactness ensures that every open cover has a finite subcover, but any finite cover is further refinable into a finite partition by zero-dimensionality. This establishes the desired ultraparacompactness.

A.6 Proof of lemma 4.7

Each open $\hat{x} = \lambda y. \bigvee \{\, b \mid x \equiv_b y \,\}$ is a $B_{\mathcal{J}}$-set homomorphism since if $y_1 \equiv_c y_2$ then

$$c \wedge \hat{x}(y_1) = \bigvee \{\, c \wedge b \mid x \equiv_b y_1 \,\} \overset{\diamond}{=} \bigvee \{\, c \wedge b \mid x \equiv_{c \wedge b} y_1 \,\} = \bigvee \{\, c \wedge b \mid x \equiv_{c \wedge b} y_2 \,\} = c \wedge \hat{x}(y_2),$$

where $\diamond$ holds from right-to-left because we can take the b on the LHS to be the $b \wedge c$ on the RHS. The assignment $x \mapsto \hat{x}$ is injective, as we now show. Let $x, y \in |F|$ and assume that $\hat{x} = \hat{y}$. Then we have $\bigvee_b \{\, b \mid x \equiv_b y \,\} = \hat{x}(y) = \hat{y}(y) = \top$, which implies there is a (cover P, WLOG refinable into a) partition P such that for each $b \in P$ we have $x \equiv_b y$. But then $y = P(\lambda b.y) = P(\lambda b.b(x, y)) = P(\lambda b.x) = x$.

Finally, let us show that $w = \bigvee_{x \in |F|} \hat{x} \wedge \mathsf{const}_{w(x)}$. Take an arbitrary $y \in |F|$. Then $w(y) = \top \wedge w(y) = \hat{y}(y) \wedge w(y) \leq \bigvee_{x \in |F|} \hat{x}(y) \wedge \mathsf{const}_{w(x)}(y)$. On the other hand, for the right-to-left inequality, it suffices to show for any $b \in B$ with $x \equiv_b y$ that $b \wedge w(x) \leq w(y)$. But this is clearly true since $x \equiv_b y$ implies $w(x) \equiv_b w(y) \iff b \wedge w(x) = b \wedge w(y)$.

A.7 Proof of proposition 4.8

The map $\sigma \colon E(F) \to L$ is a local homeomorphism by taking the family $\{\, \hat{x} \,\}_{x \in |F|}$, which is covering by 4.7. Each such open $\hat{x}$ is homeomorphic onto the whole base space $\mathcal{O}L$.

The set of global sections to $E(F) \to L$ forms a $B_{\mathcal{J}}$-set, $s_1 \equiv_b s_2 \iff \forall w \in \mathcal{O}E(F).s_1(w) \wedge b = s_2(w) \wedge b$. Every element $x \in |F|$ corresponds to a global section $s_x \colon L \to E(F)$ given by $s_x^{-1} \colon w \mapsto w(x)$. This defines a $B_{\mathcal{J}}$-set homomorphism since $x \equiv_b y$ implies, for all $w \in \mathcal{O}E$ that $s_y(w) \wedge b = w(y) \wedge b = w(b(x, y)) \wedge b = (b \wedge w(x) \vee \neg b \wedge w(y)) \wedge b = b \wedge w(x)$, i.e., that $s_x \equiv_b s_y$. Moreover, this assignment is *injective* by the injectivity of $x \mapsto \hat{x}$ (lemma 4.7).

On the other hand, to see that this assignment is *surjective*, notice that a global section s given by a frame homomorphism $s^{-1}\colon \mathcal{O}E(F) \to \mathcal{O}L$ induces a map $C\colon x \mapsto s^{-1}(\hat{x})\colon |F| \to \mathcal{O}L$. The image of this map covers $\mathcal{O}L$ because when we take all the opens $\hat{x}$ together, they cover $\mathcal{O}E(F)$. Since L is ultraparacompact, we can refine C to an (extended) partition $P\colon |F| \to \mathcal{O}L$ (not uniquely, but we don't care which one we choose). This yields an element $p \in |F|$ by taking the amalgamation $p := P[|F|]^-(\lambda b.P^{-1}b)$. The element p induces the section s_p, so to see that $s_p = s$:

$$\begin{aligned} s_p^{-1}(w) = w(p) &= w(P[|F|]^-(\lambda b.P^{-1}b)) \\ &= \bigvee_{b\in P} b \wedge w(P^{-1}b) = \bigvee_{x\in|F|} P(x) \wedge w(x) \\ &= \bigvee_{x\in|F|} C(x) \wedge w(x) = \bigvee_{x\in|F|} s^{-1}(\hat{x}) \wedge w(x) = s^{-1}(w) \end{aligned}$$

whereby the last equality follows from lemma 4.7.

A.8 Proof of proposition 4.9

Given a germ $[x]_p$, we can define a point $q \in \mathsf{pt}\, E(F) \cong \mathsf{Set}_{B_\mathcal{J}}(F, L) \to \mathcal{O}1$ by letting $q(w) = \top$ iff $p \in w(x)$. This is coherent with respect to the choice of x, because if $x \equiv_p y$ then $x \equiv_b y$ for some $b \ni p$, so $w(x) \equiv_b w(y)$ which entails $b \wedge w(x) = b \wedge w(y)$ by definition. But then $p \in w(x) \iff p \in b \wedge w(x) \iff p \in b \wedge w(y) \iff p \in w(y)$.

On the other hand, suppose we have a point $q\colon 1 \to E(F)$. This defines a subset $\hat{q} \subseteq |F| = \{\, x \mid q \in \hat{x} \,\}$, which has to be non-empty because otherwise $q \notin \hat{x}$ for all $x \in |F|$, and hence $q^{-1}(\top) = \bigvee_{x\in|F|} q^{-1}(\hat{x}) = \bot$ contradicting q^{-1} being a frame homomorphism. q also induces a point $p := \sigma q$, and the germ induced by q is $[x]_p$ for any $x \in \hat{q}$. This is coherent: for any $x, y \in \hat{q}$ we have $q \in \hat{x} \wedge \hat{y}$, but by lemma 4.7 there is some $z \in |F|$ such that $q \in \hat{z}$ and $p \in (\hat{x} \wedge \hat{y})(z)$. Then

$$\begin{aligned} p \in (\hat{x} \wedge \hat{y})(z) &\iff p \in \{\, b \wedge b' \mid x \equiv_b z, z \equiv_{b'} y \,\} \\ &\implies \exists b \wedge b' \ni p. x \equiv_{b\wedge b'} z \text{ and } z \equiv_{b\wedge b'} y \\ &\implies \exists b'' \ni p. x \equiv''_b y \\ &\iff x \equiv_p y \end{aligned}$$

From here it is a routine unfolding of definitions to see that a germ induces itself by going back-and-forth. On the other hand, given a point q, going forth-and-back produces point q' with $q' \in w$ iff $q \in \mathsf{const}_{w(x)}$ for some $x \in \hat{q}$ iff $q \in \bigvee_x \hat{x} \wedge \mathsf{const}_{w(x)} = w$, and hence $q' = q$.

The subbasic opens $[x|b]$ on the set of germs is induced by the opens $\hat{x} \wedge \mathsf{const}_b$, which generate all other opens by lemma 4.7.

A.9 Proof of lemma 4.13

First, for self-containedness we lay out precisely the definition of $B_\mathcal{J}$-congruence.

Definition A.2 Let X be a $B_\mathcal{J}$-set. An equivalence relation $\approx\, \subseteq X \times X$ is a $B_\mathcal{J}$-set congruence if for every partition $P \in \mathcal{J}$, and two families $x, x'\colon P \to X$ such that for each b, $x_b \approx x'_b$, we have

$$P(x)\ R\ P(x').$$

Given a set of pairs $G \subseteq X \times X$, The congruence $\approx_G$ generated by G is given by the following inference

rules

$$\frac{(x_1, x_2) \in G}{x_1 \approx_G x_2}\ \text{GEN} \qquad \frac{x \in X}{x \approx_G x}\ \text{REFL} \qquad \frac{x_1 \approx_G x_2 \qquad x_2 \approx_G x_3}{x_1 \approx_G x_3}\ \text{TRANS} \qquad \frac{x_1 \approx_G x_2}{x_2 \approx_G x_1}\ \text{SYMM}$$

$$\frac{x_b \approx_G x'_b \quad \forall b \in P}{P(x) \approx_G P(x')}\ \text{CONG-}P$$

We also define $\rightsquigarrow_G$ as the relation derivable by exactly one use of GEN, any use of CONG-P for any *finite* partition P, and any use of REFL.

If $\mathcal{J}$ only contains finite partitions, then the algebraic theory of $B_{\mathcal{J}}$-sets only has finite operations and we can easily show that $\approx_G$ is the symmetric transitive closure of $\rightsquigarrow_G$. However, if $\mathcal{J}$ has infinite partitions, then this is no longer the case, but we can still prove that $\approx_G$ is always derivable by one congruence applied to $\leftrightsquigarrow_G^\omega$, i.e., the transitive symmetric closure of the relation $\rightsquigarrow_G$.

Lemma A.3 *If $x_1 \rightsquigarrow_G x_2$, then this can be derived with exactly one application of the* CONG *rule.*

Proof. By induction on derivation of $x_1 \rightsquigarrow_G x_2$. In the base case, we clearly have zero applications, but we can add an application of the CONG rule over the one-element partition $\{ \top \}$. In the inductive case, we have a derivation which looks like

$$\frac{x_b \rightsquigarrow_G y_b \quad \forall b \in P}{P(x) \rightsquigarrow_G P(y)}.$$

By the inductive hypothesis, each subderivation of $x_b \rightsquigarrow_G y_b$ can be rewritten to use exactly one CONG rule, so each subderivation is associated with a partition Q_b, defining a map $Q \colon P \to \mathcal{J}$. The derivation now looks like the derivation on the left-hand side below, which is derivable as on the right-hand side.

$$\cfrac{\cfrac{\cfrac{\cdots}{x_b^c \rightsquigarrow_G y_b^c}\ \text{GEN OR REFL} \quad \forall c \in Q_b}{x_b = Q_b(\lambda c.x_b^c) \rightsquigarrow_G Q_b(\lambda c.y_b^c) = y_b}\ \text{CONG} \quad \forall b \in P}{P(x) \rightsquigarrow_G P(y)}\ \text{CONG} \qquad \cfrac{\cfrac{\cdots}{x_b^c \rightsquigarrow_G y_b^c}\ \text{GEN OR REFL} \quad \forall (b \wedge c) \in P; Q}{P; Q(\lambda b \wedge c.x_b^c) \rightsquigarrow_G P; Q(\lambda b \wedge c.y_b^c)}\ \text{CONG}$$

The right-hand derivation uses only one CONG, concluding the proof. □

Lemma A.4 *If $x_1 \approx_G x_2$ then this is derivable by a derivation of the form*

$$\cfrac{\cfrac{\vdots}{x_b \leftrightsquigarrow_G^\omega y_b} \quad \forall b \in P}{x_1 = P(x) \approx P(y) = x_2}\ \textit{CONG}$$

Proof. By induction on the derivation of $x_1 \approx_G x_2$. The base cases and the inductive cases for SYMM and CONG-P are easy, so we only work out the case for TRANS. By induction hypothesis we know that our derivation will look like

$$\cfrac{\cfrac{\cfrac{\vdots\,\Delta_b^x}{x_b \leftrightsquigarrow_G^\omega x'_b} \quad \forall b \in P}{P(x) \approx_G P(x')}\ \text{CONG} \qquad \cfrac{\cfrac{\vdots\,\Delta_c^y}{y'_c \leftrightsquigarrow_G^\omega y_c} \quad \forall c \in Q}{Q(y') \approx_G Q(y)}\ \text{CONG} \qquad P(x') = Q(y')}{x_1 = P(x) \approx_G Q(y) = x_2}\ \text{TRANS}$$

We can re-arrange this into the following derivation,

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\,\Delta_b^x}{x_b \leftrightsquigarrow_G^\omega x'_b} \quad \dfrac{}{* \approx_G *}\,\text{REFL}}{(b\wedge c)(x_b,*) \approx_G (b\wedge c)(x'_b,*)}\,\text{CONG} \quad \dfrac{\dfrac{\vdots\,\Delta_c^y}{y'_c \leftrightsquigarrow_G^\omega y_c} \quad \dfrac{}{* \approx_G *}\,\text{REFL}}{(b\wedge c)(y'_c,*) \approx_G (b\wedge c)(y_c,*)}\,\text{CONG} \qquad (b\wedge c)(x'_b,*) = (b\wedge c)(y'_c,*)}{(b\wedge c)(x_b,*) \approx_G (b\wedge c)(y_c,*) \qquad \forall b\wedge c \in P;Q}\,\text{TRANS}}{P;Q(\lambda b\wedge c.(b\wedge c)(x_b,*) \approx_G P;Q(\lambda b\wedge c.(b\wedge c)(y_c,*)}\,\text{CONG}$$

where $*$ is allowed to be any element of the $B_\mathcal{J}$-set X (if X is empty the theorem is vacuously true anyway). Here, the equality $(b\wedge c)(x'_b,*) = (b\wedge c)(y'_c,*)$ follows from $P(x') = Q(y')$, since $(b\wedge c)(P(x'),*) = c(b(P(x'),*),*) = c(b(b(P(x'),x'_b),*),*) = c(b(x'_b,*),*) = (b\wedge c)(x'_b,*)$ and similarly $(b\wedge c)(Q(y'),*) = (b\wedge c)(y'_c,*)$.

Now, we see on the lefthand-side that $P(x) = P;Q(\lambda b\wedge c.x_b) = P;Q(\lambda b\wedge c.(b\wedge c)(x_b,*))$, and similarly for $Q(y)$ on the righthand-side. Each of the derivations of $(b\wedge c)(x_b,*) \approx_G (b\wedge c)(y_c,*)$ only uses finite congruences, so can be re-arranged into derivations of $(b\wedge c)(x_b,*) \leftrightsquigarrow_G^\omega (b\wedge c)(y_c,*)$, which concludes this inductive case and hence the proof. □

Now let G be the generating equation of definition 4.10, and for which we will omit the subscript from this point on. The proof of the actual lemma proceeds in two steps. We first prove lemma A.5, which is the version of lemma 4.13 for $\rightsquigarrow_G^\omega$ (which actually suffices to prove lemma 4.13 in the case where $\mathcal{J}$ is finitary), and then prove the statement for $\approx$.

Lemma A.5 *For any $x,y \in T1[B]^\mathcal{J}$, if $x \leftrightsquigarrow^\omega y$ then for each $m,n \in T1$, we have $m \sim_{x(m)\wedge y(n)} n$.*

Proof. A derivation of $x \leftrightsquigarrow^\omega y$ is a (composable) chain of either $\rightsquigarrow$ or $\leftsquigarrow$ derivations, e.g. $x = x_0 \rightsquigarrow x_1 \leftsquigarrow x_2 \rightsquigarrow x_3 \rightsquigarrow x_4 \leftsquigarrow \ldots \rightsquigarrow x_k = y$ (arrow directions non-indicative). Each such derivation $x_i \rightsquigarrow x_{i+1}$, by lemma A.3, can be rewritten with exactly one CONG rule over an associated partition R. This means that the derivation looks like

$$\dfrac{\vdots}{x_i = R(h) \rightsquigarrow R(h') = x_{i+1}}$$

where for a unique $b_0 \in R$, we have $h_{b_0} = t \gg\!= u$ and $h'_{b_0} = P^{(t)}(\lambda a.t \gg u(a))$ and for $b \neq b_0 \in R$, we have $h_b = h'_b$. Associate to x_i the partition $P_i^{\leftarrow} := R$, and to x_{i+1} the partition

$$P_{i+1}^{\rightarrow} := R \gg\!= \lambda b. \begin{cases} P^{(t)} & \text{if } b = b_0 \\ \{\top\} & \text{otherwise.} \end{cases}$$

For a derivation $x_i \leftsquigarrow x_{i+1}$, perform the opposite assignment. So each x_i is then associated with two partitions $P_i^{\rightarrow}$ and $P_i^{\leftarrow}$, except for x_0 and x_k. For these, define $P_0^{\rightarrow} := P$ and $P_k^{\leftarrow} := Q$. Since there are finitely many of these partitions, we can take a common refinement—call this S.

Consider $d \in S$. It refines a unique $b \in P$, which identifies the term $t_d^0 := t_b$. Now look at the first derivation, and suppose it is $x_0 \rightsquigarrow x_1$. Then d refines a unique $c \in P_1^{\rightarrow}$. There are two possible cases:

(i) Either $c \in P_0^{\leftarrow}$, in which case we define $t_d^1 := t_d^0$;

(ii) or $c = c_0 \wedge [t \mapsto a]$ for some $c_0 \in P_0^{\leftarrow}$, in which case we know that t_d^0 must be of the form $t \gg\!= u$. So define $t_d^1 = t \gg u(a)$.

We note that in either case, we have $t_d^0 \sim_d t_d^1$.

The other possibility is that $x_0 \leftsquigarrow x_1$. Then d refines a unique $c \in P_0^{\leftarrow}$. There are two possible cases:

(i) Either $c \in P_1^{\rightarrow}$, in which case we define $t_d^1 := t_d^0$;

(ii) or $c = c_0 \wedge [t \mapsto a]$ for some $c_0 \in P_1^{\rightarrow}$, in which case we know that t_d^0 must be of the form $t \gg u(a)$. So define $t_d^1 = t \gg\!= u$.

Now we may repeat this process, obtaining $t_b = t_d^0 \sim_d t_d^1 \sim_d \ldots \sim_d t_d^k$. Here, since d refines some $c \in Q$, we have that $t_d^k = s_c$. So we may conclude $t_b \sim_d s_c$. To finish the proof, we see that any $b \wedge c \in P \wedge Q$ is a join of its refinements in S, and since we show $t_b \sim_d s_c$ for all of its refinements d, we can conclude that $t_b \sim_{b\wedge c} s_c$. □

Finally, we prove lemma 4.13.

($\implies$) suppose $x_1 \approx x_2 \in T1[B]^{\mathcal{J}}$. Then by lemma A.4, we know that $x_1 = P(x) \approx P(y) = x_2$ for some partition $P \in \mathcal{J}$ and families $x, y \colon P \to T1[B]^{\mathcal{J}}$ such that for each $b \in P$, $x_b \leftrightsquigarrow^{\omega} y_b$. So by lemma A.5, we have $m \sim_{x_b(m)\wedge y_b(n)} n$ for every $m, n \in T1$. Now, recall from 4.3 that $P(x)(m) := \bigvee_{b\in P} b \wedge x_b(m)$, so $P(x)(m) \wedge P(y)(n) = \bigvee_{b\in P} b \wedge x_b(m) \wedge y_b(n)$. Hence $P(x)(m) \wedge P(y)(n) \leq [\![m \sim n]\!]$ iff for all $b \in P$, $b \wedge x_b(m) \wedge y_b(n) \leq [\![m \sim n]\!]$, which we have.

($\impliedby$) Suppose $m \sim_{x_1(m)\wedge x_2(n)} n$ for each $m, n \in T1$. Let $P = \{\, x_1(m) \wedge x_2(n) \mid m, n \in T1 \,\}^-$ be the common refinement of x_1 and x_2. Abusing notation, we will write families indexed by elements of P as indexed by pairs (m, n), for example $\lambda(m, n).x_{(m,n)}$. Then $x_1 = P(\lambda(m, n).m)$ and $x_2 = P(\lambda(m, n).n)$, and so by CONG-P it suffices to prove $b(m, n) \approx n$ for each $m, n \in T1$ and $b \in P$ with $m \sim_b n$.

Consider first the special case where $b \leq [\![m \sim_1 n]\!]$. Then

$$\left\{ [t \mapsto a] \;\middle|\; \begin{array}{c} A \in \mathsf{Set}, |A| \leq \kappa, t : TA, u, v \colon A \to T1, a \in A, \\ u(a) = v(a), m = t \gg\!= u, n = t \gg\!= v \end{array} \right\} \cup \{\, \neg b \,\}$$

is an open cover, so there is a refining partition P. We can further refine this partition to $Q = P; \{\, b, \neg b \,\}$. Now each $q \in Q$ is either $q \leq \neg b$, or $q \leq b$ and associated with some $t_q \in TA$, $a_q \in A$ and families u_q, v_q such that $q \leq [t_q \mapsto a_q]$, $u_q(a) = v_q(a)$, $m = t_q \gg\!= u_q$ and $n = t_q \gg\!= v_q$. Then we can derive

$$\begin{aligned}
b(m, n) &= Q\left(\lambda q. \begin{cases} m & q \leq b \\ n & q \leq \neg b \end{cases}\right) \\
&= Q\left(\lambda q. \begin{cases} t_q \gg\!= u_q & q \leq b \\ n & q \leq \neg b \end{cases}\right) \\
&\approx Q\left(\lambda q. \begin{cases} P^{(t_q)}(\lambda a. t_q \gg u_q(a)) & q \leq b \\ n & q \leq \neg b \end{cases}\right) && \text{by definition of } \approx \\
&= Q\left(\lambda q. \begin{cases} t_q \gg u_q(a_q) & q \leq b \\ n & q \leq \neg b \end{cases}\right) && \text{since } q \leq [t_q \mapsto a_q] \\
&= Q\left(\lambda q. \begin{cases} t_q \gg v_q(a_q) & q \leq b \\ n & q \leq \neg b \end{cases}\right) \\
&= b(n, n) = n && \text{by similar reasoning}
\end{aligned}$$

Now, consider the general case: by ultraparacompactness, it suffices to consider when $b \leq [\![m \sim_k n]\!]$ for each $k \in \mathbb{N}$. Then $\{\, \bigwedge_{i=1}^{k-1} [\![m_i \sim_1 m_{i+1}]\!] \mid m_1 = m, m_2 \ldots m_{k-1} \in T1, m_k = n \,\} \cup \{\, \neg b \,\}$ is refinable by a partition Q such that each $q \in P$ is either $q \leq \neg b$ or $q \leq b$ and $q \leq \bigwedge_{i=1}^{k-1} [\![m_i \sim_1 m_{i+1}]\!]$ for some $\{\, m_i \,\}_{i\leq k}$. Then we can prove $q(m_i, n) \approx q(m_{i+1}, n)$ by similar reasoning as the previous paragraph, so by transitivity

of $\approx$ we have $q(m,n) \approx q(n,n) = n$. Then we finally finish the proof with the following equational reasoning:

$$\begin{aligned} b(m,n) &= Q\left(\lambda q. \begin{Bmatrix} m & q \leq b \\ n & q \leq \neg b \end{Bmatrix}\right) \\ &= Q\left(\lambda q. \begin{Bmatrix} q(m,n) & q \leq b \\ n & q \leq \neg b \end{Bmatrix}\right) \\ &\approx Q\left(\lambda q. \begin{Bmatrix} q(n,n) & q \leq b \\ n & q \leq \neg b \end{Bmatrix}\right) \\ &= b(n,n) = n. \end{aligned}$$

A.10 Proof that definition 4.14 corresponds to definition 4.10

Suppose $w\colon T1 \to \mathsf{LB}_0 T$ is a function respecting trace equivalence as in 4.14. Then we need to show $w(t \gg\!= u) = P^{(t)}(\lambda[t \mapsto a].t \gg u(a))$, which we have by

$$\begin{aligned} w(t \gg\!= u) &= P^{(t)}(\lambda[t \mapsto a].w(t \gg\!= u)) && (2) \\ &= P^{(t)}(\lambda[t \mapsto a].[t \mapsto a](w(t \gg\!= u), w(t \gg u(a)))) && (2) \\ &= P^{(t)}(\lambda[t \mapsto a].w(t \gg u(a))) && (*) \end{aligned}$$

where (*) follows because w respects trace equivalence:

$$\begin{aligned} & t \gg\!= u \sim_{[t\mapsto a]} t \gg u(a) \\ \implies & w(t \gg\!= u) \equiv_{[t\mapsto a]} w(t \gg u(a)) \\ \iff & [t \mapsto a](w(t \gg\!= u), w(t \gg u(a))) = w(t \gg u(a)) \end{aligned}$$

On the other hand, if $\tilde{w}\colon F_T \to \mathsf{LB}_0 T$ is a $B_\mathcal{J}$-set homomorphism, then we need to show $\tilde{w}$ restricts to a function $w\colon T1 \to \mathsf{LB}_0 T$ which respects trace equivalence. Consider then two trace equivalent terms $m \sim_b n \in T1$. Then we have $\tilde{w}(m) \equiv_b \tilde{w}_n$ since

$$b(\tilde{w}(m), \tilde{w}(n)) = \tilde{w}(b(m,n)) = \tilde{w}(n)$$

where the last equality follows because $b(m,n) = n \iff m \sim_b n$ by lemma 4.13.

A.11 Proof of proposition 4.16

By proposition 4.9, we know that $\mathsf{pt}(\mathsf{LB}_1 T) \cong \Sigma_{\beta \in \mathsf{pt}\,\mathsf{LB}_0 T} F_T/_{\equiv_\beta}$. But we know $\mathsf{pt}\,\mathsf{LB}_0 T \cong \mathbb{B}_0 T$, so the β really are just admissible behaviours. Next, observe that every $x \in F_T$ can be expressed in the form $x = P(\lambda b.m_b)$, and hence $x \equiv_b m_b$ for the $b \in P$ with $\beta \in b$. Hence, we have $F_T/_{\equiv_b} \cong T1/_{\equiv_\beta} \cong T1/_{\sim_\beta}$ over each β. Therefore $\mathsf{pt}(\mathsf{LB}_1 T) \cong \Sigma_{\beta \in \mathbb{B}_0 T} T_1/_{\sim_\beta} = \mathbb{B}_1 T$.

A.12 Proof of lemma 4.17

The following decomposition lemma, analogous to lemma 4.15, will come in handy.

Lemma A.6 *Every $h\colon T1 \to L$ with the conditions of this lemma is of the form $h = \bigvee_m m^* \wedge \mathsf{const}_{h(m)}$ where $m^* := \lambda n. f^{-1}[\![m \sim n]\!]$.*

Proof. We have to show $h(n) = \bigvee_m f^{-1}[\![m \sim n]\!] \wedge h(m)$. As in the proof of lemma 4.15, the left-to-right inequality is easy, so we focus on the right-to-left inequality for which we have to show $f^{-1}[\![m \sim n]\!] \wedge h(m) \leq h(n)$. By ultraparacompactness, $f^{-1}[\![m \sim n]\!] = \bigvee \{\, f^{-1}b \mid b \leq [\![m \sim n]\!], b \in B \,\}$ so it suffices to prove for each complemented $b \leq [\![m \sim n]\!]$ that $f^{-1}b \wedge h(m) \leq h(n)$, but this immediately follows from the condition on h.

It is easy to see that $\mathsf{const}_{h(m)}$ satisfies the condition since it is just a constant map. Meanwhile, for m^* whenever $n_1 \sim_b n_2$ we have that

$$m^* n_1 \wedge f^{-1} b = f^{-1}[\![m \sim n_1]\!] \wedge f^{-1} b = f^{-1}([\![m \sim n_1]\!] \wedge b) = f^{-1}([\![m \sim n_2]\!] \wedge b) m^* n_2 \wedge f^* b.$$

Hence m^* satisfies the condition of this lemma. □

The two projections $\pi_1 \colon L \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \to L$ and $\pi_2 \colon L \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \to \mathsf{LB}_1 T$ are given by $\pi_1^{-1} \colon u \mapsto \mathsf{const}_u$ and $\pi_2^{-1} \colon w \mapsto \lambda m. f^{-1} w(m)$. Given a pullback cone $i \colon Z \to L$ and $j \colon Z \to \mathsf{LB}_1 T$, if a universal arrow $\langle i, j \rangle$ exists then it must satisfy

$$\langle i, j \rangle^{-1} (\mathsf{const}_{h(m)}) = \langle i, j \rangle^{-1} \pi_1^{-1} h(m) = i^{-1} h(m)$$

$$\langle i, j \rangle^{-1} (m^*) = \langle i, j \rangle^{-1} \pi_2^{-1} h(\hat{m}) = j^{-1} \hat{m}$$

But then by lemma A.6, the frame of opens for the pullback is generated by these opens, so these two equations uniquely determine $\langle i, j \rangle$. It is then straightforward to check that this is well-defined.

A.13 Proof that definition 4.18 is a localic category

It is very straightforward to check that the domain and codomain of identity and compositions correspond to what they should be, so we focus on the unitality and associativity. It is easy to see from lemma 4.17 that $\mathsf{LB}_0 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \cong \mathsf{LB}_1 T$, for which the unitality diagram becomes

$$\begin{array}{ccccc} \mathsf{LB}_1 T & \xrightarrow{h \mapsto h(\mathsf{return}, -)} & \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T & \xleftarrow{h \mapsto h(-, \mathsf{return})} & \mathsf{LB}_1 T \\ & \searrow & \downarrow{\scriptstyle w \mapsto w(- \gg -)} & \swarrow & \\ & & \mathsf{LB}_1 T & & \end{array}$$

and this commutes by unitality of $\gg$. Finally, by lemma 4.17 the pullback of composable triples $\mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T$ can be constructed as the frame of appropriate maps $T1 \times T1 \times T1 \to \mathcal{O}\mathsf{LB}_0 T$, for which the associativity diagram below obviously commutes due to the associativity of $\gg$.

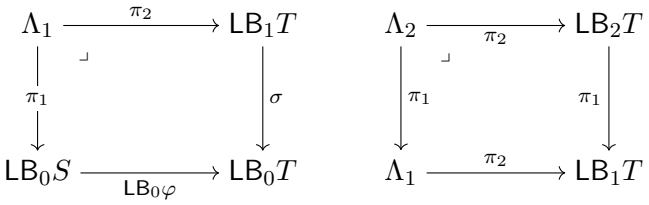
$$\begin{array}{ccc} \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T & \xrightarrow{h \mapsto h(-, - \gg -)} & \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \\ \downarrow{\scriptstyle h \mapsto h(- \gg -, -)} & & \downarrow{\scriptstyle w \mapsto w(- \gg -)} \\ \mathsf{LB}_1 T \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T & \xrightarrow[w \mapsto w(- \gg -)]{} & \mathsf{LB}_1 T \end{array}$$

A.14 Proof of lemma 4.20

Each $\beta \in \mathbb{B}_0 T$ admits an open neighborhood $s^{-1}[m_\beta | \, \mathsf{return}\, 1]$ where t_β is some representative of the equivalence class $s(\beta)$. This induces an open cover on $\mathbb{B}_0 T$ which is refined by a finite partition $\{ b_i \}_{i \in I}$ since $\mathbb{B}_0 T$ is a Stone space. It suffices to pick m_i to be the m_β of an open $s^{-1}[m_\beta | \, \mathsf{return}\, 1]$ refined by b_i. The uniqueness under trace equivalence is easy to see because for any $\beta \in b_i \wedge b_j$, we have $[m_i]_\beta = s(\beta) = [m_j]_\beta$. The spatiality of $\mathsf{LB}_0 T$ then ensures this corresponds to $m_i \sim_{b_i \wedge b_j} m_j$.

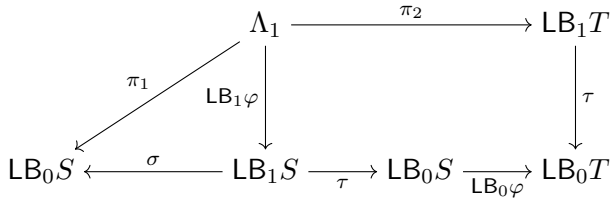
A.15 Details to the proof of proposition 5.2

First we have to verify that $\mathsf{LB}\varphi$ is an internal retrofunctor. For this we need to consider the pullbacks

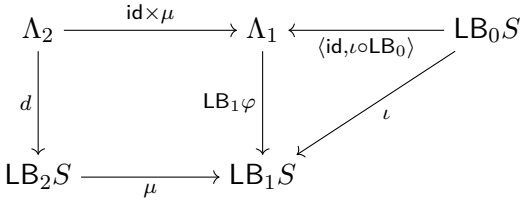


Noting that Λ_2 is the pullback of the second square composed with the first square, and following similar ideas to lemma 4.17, the second pullback Λ_2 can be expressed by the frame of maps $h : T1 \times T1 \to \mathcal{O}\mathsf{LB}_0 S$ such that $m \sim_b m'$ implies $h(m,n) \wedge (\mathsf{LB}_0\varphi)^{-1} b = h(m,n) \wedge (\mathsf{LB}_0\varphi)^{-1} b$, and $n \sim_b n'$ implies $h(m,n) \wedge (\mathsf{LB}_0\varphi)^{-1}(\!|m|\!)^{-1} b = h(m,n') \wedge (\mathsf{LB}_0\varphi)^{-1}(\!|m|\!)^{-1} b$.

The requirements on the domain and codomain of the lift is encoded by requiring the following diagram to commute:



but this follows by a straightforward chase along the diagram. Next, to see that identity and composition is respected, we require the following diagrams to commute:



For the commutativity involving ι, this amounts to checking, for $w \in \mathcal{O}\mathsf{LB}_1 S$, that

$$\bigvee_{m \in T1} w(\varphi(m)) \wedge (\mathsf{LB}_0\varphi)^{-1}[\![m \sim \mathsf{return}]\!] = w(\mathsf{return}).$$

The proof of this is similar to the proof of lemma A.6. For the commutativity involving μ, the map d is defined on inverse image by $d^{-1} : h \mapsto h(\varphi_1(-), \varphi_1)$. Then the commutativity of this square amounts to checking, for $w \in \mathcal{O}\mathsf{LB}_1 T$, that $w(\varphi(-) \gg \varphi(-)) = w(\varphi(- \gg -))$ but this easily follows from φ being a monad map.

We now also must verify that LB is functorial. If $\varphi = \mathsf{id} : T \to T$ then we see that the definition of $\mathsf{LB}\varphi$ is indeed the identity retrofunctor. For composition, given $\varphi : T \to S$ and $\psi : S \to R$, it is obvious that $\mathsf{LB}_0(\varphi \circ \psi) = \mathsf{LB}_0(\psi) \circ \mathsf{LB}_0(\varphi)$. For the action on morphisms, the composite is given by

$$\mathsf{LB}_0 R \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \xrightarrow{\langle \pi_0, \mathsf{LB}_1\varphi \circ (\mathsf{LB}_0\psi \times \mathsf{id})\rangle} \mathsf{LB}_0 R \times_{\mathsf{LB}_0 S} \mathsf{LB}_1 S \xrightarrow{\mathsf{LB}_1\psi} \mathsf{LB}_1 R$$

Let us compute the inverse image of an open set $w \in \mathcal{O}(\mathsf{LB}_1 R)$ along this map. The inverse along $\mathsf{LB}_1\psi$ gives $w \circ \psi_1$ which by lemma A.6 can be expressed as $\bigvee_{s \in S1} s^* \wedge \mathsf{const}_{w(\psi(s))}$. Now the inverse of $\mathsf{const}_{w(\psi(s))}$ along the pair of maps can be computed as the inverse of just the left component, which again gives $\mathsf{const}_{w(\psi(s))}$, but this time as an open in $\mathsf{LB}_0 R \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T$. The inverse of s^* along the pair can be computed as the inverse of $\hat{s}$ along the right component, which gives $\lambda t \in T1.(\mathsf{LB}_0\psi)^{-1}[\![s \sim \varphi(t)]\!]$. So

combining the two, in the end we get an open of $\mathsf{LB}_0 R \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T$ defined by

$$\lambda t \in T1.\ \bigvee_{s \in S1} (\mathsf{LB}_0\psi)^{-1}[\![s \sim \varphi(t)]\!] \wedge w(\psi(s))$$

and we have to show this is equal to $(\mathsf{LB}_1(\psi \circ \varphi))^{-1} w = w \circ \psi_1 \circ \phi_1$. But this is again the same type of reasoning as in the proof of lemma A.6.

A.16 Details to the proof of theorem 5.4

Let us write Γ as shorthand for $\Gamma\mathsf{LC}$. It suffices to prove, for any retrofunctor $F\colon \mathsf{LC} \to \mathsf{LB}T$, there is a unique monad morphism $\varphi\colon T \to \Gamma$ such that $\mathsf{LB}\varphi \circ \varepsilon = F$. For this, we show that this condition uniquely determines φ. So consider $t \in TA$ and observe that $\varphi(t)^{-1} \langle a \mapsto w\rangle = \varphi(t)^{-1} \langle a \mapsto \top\rangle \wedge \varphi(t \gg \mathsf{return})^{-1} w$. We then show that (i) $F_0^{-1}[t \mapsto a] = \varphi(t)^{-1} \langle a \mapsto \top\rangle$; and (ii) $(F_1^{-1} w)(t \gg \mathsf{return}) = \varphi(t \gg \mathsf{return})^{-1} w$. This fully determines $\varphi(t)^{-1} \langle a \mapsto w\rangle$ as $F_0^{-1}[t \mapsto a] \wedge (F_1^{-1} w)(t \gg \mathsf{return})$.

Now, (i) is a straightforward unfolding of definitions on the equation $F_0^{-1} = \varepsilon_0^{-1}(\mathsf{LB}_0\varphi)^{-1}$, so we leave this as an exercise to the reader (if the reader is still reading). For (ii), we have

$$F_1 = \mathsf{LC}_0 \times_{\mathsf{LB}_0 T} \mathsf{LB}_1 T \xrightarrow{\langle \pi_0, \mathsf{LB}_1\varphi \circ (\varepsilon_0 \times \mathsf{id})\rangle} \mathsf{LC}_0 \times_{\mathsf{LB}_0 \Gamma} \mathsf{LB}_1 \Gamma \xrightarrow{\varepsilon_1} \mathsf{LC}_1$$

Let us compute the inverse image of $w \in \mathsf{LC}_1$ along this map. First, by lemma A.6 we can decompose $\varepsilon_1^{-1} w = \bigvee_{m \in \Gamma 1} \mathsf{const}_{m^{-1} w} \wedge m^*$. Then the the inverse image of $\mathsf{const}_{m^{-1}w}$ along the pair is given simply as $\mathsf{const}_{m^{-1}w}$ while the inverse image of m^* is $\lambda n \in T1. \varepsilon_0^{-1}[\![\varphi(n) \sim m]\!]$. Therefore, we arrive at the result

$$F_1^{-1} w = \bigvee_m \lambda n \in T1. m^{-1} w \wedge \varepsilon_0^{-1}[\![\varphi(n) \sim m]\!]$$

Hence, if we let $n := t \gg \mathsf{return}$, then we have $F_1^{-1} w(n) = \bigvee_{m \in \Gamma 1} m^{-1} w \wedge \varepsilon^{-1}[\![\varphi(n) \sim m]\!]$. Now, it is easy to see that $\varphi(n)^{-1} w \leq F_1^{-1} w(n)$ by taking $m := \varphi(n)$. On the other hand, for $\varphi(n)^{-1} w \geq F_1^{-1} w(n)$ we have to reason in terms of witnesses of $[\![\varphi(n) \sim m]\!]$. For simplicity, we simply consider a 1-step witness $[h \mapsto b]$ for $h \in \Gamma B, b \in B$ such that $\varphi(n) = h \gg\!= u$ and $m = h \gg\!= v$, with $u(b) = v(b)$. Then one can see that

$$\begin{aligned} m^{-1} w \wedge \varepsilon_0^{-1}[h \mapsto b] &= (h \gg\!= v)^{-1} w \wedge h^{-1} \langle b \mapsto \top\rangle \\ &= (h \gg v(b))^{-1} w \wedge h^{-1} \langle b \mapsto \top\rangle \\ &= (h \gg\!= u)^{-1} w \wedge h^{-1} \langle b \mapsto \top\rangle \\ &\leq \varphi(n)^{-1} w \end{aligned}$$

This proves (ii) and hence we conclude that φ is uniquely determined.

A.17 Proof of proposition 6.3

Proof. A computation reveals that $(h \gg \mathsf{return})^{-1} w = \bigvee_a h^{-1} \langle a \mapsto w\rangle$, while $\mathsf{return}^{-1} w = \iota^{-1} w$. So these two are equal iff $h \gg \mathsf{return}\, a = \mathsf{return}\, a$. Notice that we do not use the second condition of being hyperaffine, because it is automatically true for any h satisfying the first condition (also known as *affine*). Now, suppose that such a hyperaffine $\overline{s}$ for a section $s \in \Gamma\mathsf{LC}(A)$. Then a straightforward computation (using the characterization of hyperaffines) reveals that the condition $s = \overline{s} \gg\!= \lambda a. s \gg \mathsf{return}\, a$ implies $\overline{s}^{-1} \langle a \mapsto w\rangle = s^{-1} \langle a \mapsto \top\rangle \wedge \iota^{-1} w$ which determines $\overline{s}^{-1}$ as a well-defined frame homomorphism. See below for the computations. □

(affine characterization) Applying the definition of $\gg\!=$, we find that $(h \gg\!= \mathsf{return})^{-1} w = \bigvee_a h^{-1} \langle a \mapsto w' \rangle$ where $w' = \bigvee \{ v_1 \wedge \tau^{-1}\iota^{-1} v_2 \mid v_1 \times v_2 \leq \mu^{-1} w \}$. But notice that w' is the inverse image of w along

$$\mathsf{LC}_1 \xrightarrow{\langle \mathsf{id}, \tau \rangle} \mathsf{LC}_1 \times_{\mathsf{LC}_0} \mathsf{LC}_0 \xrightarrow{\mathsf{id} \times \iota} \mathsf{LC}_1 \times_{\mathsf{LC}_0} \mathsf{LC}_0 \xrightarrow{\mu} \mathsf{LC}_1$$

$$\pi_{\mathsf{LC}_1}$$

but this inverse image is equally well computed as $w \wedge \tau^{-1} \top = w$, and hence $w' = w$.

(determination of $\overline{s}$) Since $s = \overline{s}^{-1} \gg\!= \lambda a.s \gg \mathsf{return}\, a$, we can compute $s^{-1} \langle a \mapsto \top \rangle$ as

$$\overline{s}^{-1} \left\langle a \mapsto \bigvee \{ v_1 \wedge \tau^{-1} s^{-1} \bigvee_{a' \in A} \mid v_1, v_2 \in \mathcal{O}\mathsf{LC}_1 \} \right\rangle$$

Next, we know $\iota^{-1} w = \bigvee_{a'' \in A} \overline{s}^{-1} \langle a'' \mapsto w \rangle$ so

$$\iota^{-1} w \wedge s^{-1} \langle a \mapsto \top \rangle = \overline{s}^{-1} \left\langle a \mapsto w \wedge \bigvee \{ v_1 \wedge \tau^{-1} s^{-1} \bigvee_{a' \in A} \langle a' \mapsto v_2 \rangle \mid v_1, v_2 \in \mathcal{O}\mathsf{LC}_1 \} \right\rangle$$

and now we have $w \leq \bigvee \{ v_1 \wedge \tau^{-1} s^{-1} \bigvee_{a' \in A} \langle a' \mapsto v_2 \rangle \mid v_1, v_2 \in \mathcal{O}\mathsf{LC}_1 \}$ by taking $v_1 = w$ and $v_2 = \top$, so this simplifies to $\overline{s}^{-1} \langle a \mapsto w \rangle$.

A.18 Proof of lemma 6.4

The Grothendieck Boolean algebra structure $H2_{\mathcal{J}}$ is established in [11]. The Boolean algebra structure on $H2$ is given by $\top = \mathsf{return}\, 1$, $h_1 \wedge h_2 = h_1 \gg\!= (0 \mapsto \mathsf{return}\, 0; 1 \mapsto h_2)$ and $\neg h = h \gg\!= (0 \mapsto \mathsf{return}\, 1; 1 \mapsto \mathsf{return}\, 0)$, and from this it is easy to see that $H2$ satisfies (LB_0-$\bot$), (LB_0-η), $[t \gg \mathsf{return}\, a \mapsto a']$, and $\bigvee_{a \in A} [t \mapsto a] = \top$ for finite sets A. Then the only missing axioms are $\bigvee_{a \in A} [t \mapsto a] = \top$ for infinite A, but these are precisely the partitions in $\mathcal{J}$. Hence $H2_{\mathcal{J}}$ generates $\mathcal{O}(\mathsf{LB}_0 T)$.

The inverse to $\delta \colon T_1 \to F_T$ is witnessed by $\delta^{-1} \colon x \mapsto h \gg\!= \lambda b.x^{-1} b$, where $h \in HP$ is a hyperaffine realizing the partition $P^{(h)}$ induced by $x \colon T1 \to H2$. On one hand we have $\delta\delta^{-1}(x) = \delta(h \gg\!= \lambda b.x^{-1} b) = P(\delta(h \gg x^{-1} b)) = P(\delta(x^{-1} b)) = x$. On the other hand, $\delta^{-1}\delta(t) = \mathsf{return}\, \top \gg\!= \lambda b.\delta(t)^{-1} b = \delta(t)^{-1} \top = t$.

A.19 Proof of proposition 6.5

($\Longleftarrow$) Suppose now the unit map is an isomorphism. Then the hyperaffine-unary factorization of $\Gamma\mathsf{LB}T$ (proposition 6.3) must transfer along the unit map onto T.

($\Longrightarrow$) For the converse direction, we make use of lemma 6.4, which basically says any global section $s \in \Gamma\mathsf{LB}TA$ identifies a hyperaffine $h \in HA$ and a family $u : A \to T1$ of unary computations, and the composite $h \gg\!= \lambda a.u(a) \gg \mathsf{return}\, a$ induces the section s. What follows is the proof in more detail.

Assume T is hyperaffine-unary, we have to prove η_T is an isomorphism, i.e., bijective at each level. To see that η_T is surjective, consider then a section $s \in \Gamma\mathsf{LB}TA$. We can always factor $s = h \gg\!= \lambda i.\eta(m_i) \gg \mathsf{return}\, f(i)$ for some hyperaffine section $h \in HC$, some family of $T1$-terms $\{ m_i \}_{i \in I}$ and function $f \colon I \to A$. So it suffices to show that h is in the image of η_T. The data of a hyperaffine section h is completely determined by the partition $\{ h^{-1} \langle i \mapsto \top \rangle \mid i \in I \}^- \subseteq \mathsf{LB}_0 T$, but now because T is hyperaffine-unary, by lemma 6.4 such a partition has to be of the form $\{ [h' \mapsto j] \mid j \in J \}$ for some $h' \in HJ \subseteq TJ$ and $J = \{ i \in I \mid h^{-1} \langle i \mapsto \top \rangle \neq \bot \} \subseteq I$. We claim that $\eta_T(h') = h$, and this follows by unfolding definitions:

$$\eta_T(h')^{-1} \langle i \mapsto w \rangle = [h' \mapsto i] \wedge w(h' \gg \mathsf{return}) = h^{-1} \langle i \mapsto \top \rangle \wedge w(\mathsf{return}) = h^{-1} \langle i \mapsto w \rangle$$

Finally, to see that η_T is injective, consider $t_1 \neq t_2 \in TA$. Then again by proposition 6.3, both admit decompositions $t_1 = \overline{t_1} \gg\!= \lambda a.m_1 \gg \mathsf{return}\, a$ and $t_2 = \overline{t_2} \gg\!= \lambda a.m_2 \gg \mathsf{return}\, a$, so if they are not equal it

must be that either $\overline{t_1} \neq \overline{t_2} \in HA$ or $m_1 \neq m_2 \in T1$. If the former, then by hyperaffineness of h_1 and h_2:

$$\begin{aligned}
h_1 &= h_1 \gg\!= \lambda a.h_1 \gg\!= \lambda a'.\,\mathsf{return}\,a \text{ if } a = a' \text{ else } h_2 && (h_1 \text{ is h.aff.})\\
&= h_1 \gg\!= \lambda a.[h_1 \mapsto a] \gg\!= (\mathsf{return}\,a, h_2) && (\text{definition of } [h_1 \mapsto a])\\
&= h_1 \gg\!= \lambda a.[h_2 \mapsto a] \gg\!= (h_2 \gg \mathsf{return}\,a, h_2) && ([h_1 \mapsto a] = [h_2 \mapsto a] \text{ and } h_2 \text{ is h.aff.})\\
&= h_1 \gg\!= \lambda a.h_2 \gg\!= \lambda a'.h_2 \gg\!= \lambda a''.\,\mathsf{return}\,a \text{ if } a = a'' \text{ else } a'' && (\text{definition of } [h_2 \mapsto a])\\
&= h_1 \gg\!= \lambda a.h_2 \gg\!= \lambda a'.h_2 \gg\!= \lambda a''.\,\mathsf{return}\,a'' \text{ if } a' = a'' \text{ else } \ldots && (\text{the } \ldots \text{ does not matter})\\
&= (h_1 \gg\!= \lambda a.h_2) = (h_1 \gg h_2) = h_2 && (h_2 \text{ is h.aff.})
\end{aligned}$$

If the latter is true, then $\eta(m_1), \eta(m_2) \colon \mathsf{LB}_0 T \to \mathsf{LB}_1 T$, viewed as maps of local homeomorphisms, correspond to maps of sheaves $\delta(m_1), \delta(m_2) \colon 1 \to F_T$, and from the isomorphism $F_T \cong T1$ of lemma 6.4 we know these cannot be equal. It can then be verified that if we have two hyperaffines $h_1 \neq h_2 \in \Gamma\mathsf{LB}TA$ or unary sections $s_1 \neq s_2 \in \Gamma\mathsf{LB}1$ then $h_1 \gg\!= \lambda a.s_1\,\mathsf{return}\,a \neq h_2 \gg\!= \lambda a.s_2\,\mathsf{return}\,a$, and hence $\eta(t_1) \neq \eta(t_2)$.

A.20 Proof of proposition 6.7

The following lemma come in handy.

Lemma A.7 *Two internal categories are retrofunctorially isomorphic iff they are functorially isomorphic.*

Proof. Let LC and LD be internal categories. Given a functorial isomorphism $F \colon \mathsf{LC} \to \mathsf{LD}$ with inverse F^{-1}, define the retrofunctor G by $G_0 := F_0$ and $G_1 := F_1^{-1} \circ \pi \colon \mathsf{LC}_0 \times_{\mathsf{LD}_0} \mathsf{LD}_1 \to \mathsf{LD}_1 \to \mathsf{LC}_1$, and vice versa for G^{-1}. On the other hand, given retrofunctors G and G^{-1}, define the functor F by $F_0 := G_0$ and $F_1 := G_1^{-1} \circ \langle G_0\sigma, \mathsf{id}\rangle$ where $\langle G_0\sigma, \mathsf{id}\rangle : \mathsf{LC}_1 \to \mathsf{LD}_0 \times_{\mathsf{LC}_0} \mathsf{LC}_1$. Define the inverse F^{-1} similarly. We leave it to the reader to verify the necessary equations. □

For brevity, we omit the subscript LC from ε, and let us also write $\mathsf{LB}_i := \mathsf{LB}_i\Gamma\mathsf{LC}$ for $i = 0, 1$.

($\Longrightarrow$) By lemma A.7, we get an isomorphism $\mathsf{LB}_0\Gamma\mathsf{LC} \cong \mathsf{LC}_0$ so LC_0 is also ultraparacompact, and also we get an isomorphism $\mathsf{LB}_1\Gamma\mathsf{LC} \cong \mathsf{LC}_1$ commuting with the source maps, so the source map of LC is also a local homeomorphism.

($\Longleftarrow$) By lemma A.7 it suffices to prove that the counit ε partakes in a functorial isomorphism. The action on objects $\varepsilon_0 \colon \mathsf{LC}_0 \to \mathsf{LB}_0$ has inverse $\ni_0$ given on generating clopens $b \in \mathfrak{B}\mathsf{LC}_0$ by $\ni_0^{-1} \colon b \mapsto [b^+]$ where $b^+ \colon \mathsf{LC}_0 \to 2{\cdot}\mathsf{LC}_1$ given by $(b^+)^{-1} \colon \langle 1 \mapsto w\rangle \mapsto b \wedge \iota_{\mathsf{LC}}^{-1} w$ and $(b^+)^{-1} \colon \langle 0 \mapsto w\rangle \mapsto \neg b \wedge \iota_{\mathsf{LC}}^{-1} w$. This map is well-defined because it realizes all partitions of LC_0: any partition P manifests as a section $P^+ \in \Gamma\mathsf{LC}(P)$ defined analogously to b^+, and hence we have $\top = \bigvee_{b \in P} [P^+ \mapsto b] = \bigvee_{b \in P} [b^+]$. It is straightforward to see that $\varepsilon_0^{-1}\ni_0^{-1} = \mathsf{id}$. On the other hand, to see that $\ni_0^{-1}\varepsilon_0^{-1} = \mathsf{id}$, consider a generating open $[s]$ where $s \in \Gamma\mathsf{LC}2$. By proposition 6.3 we have its corresponding hyperaffine $\overline{s}$, and it is easy to see that $[\overline{s}] = [s]$. Then, it is a matter of checking that $(\varepsilon_0^{-1}[s])^+ = \overline{s}$.

This gives us an internal functor $\mathcal{E}$ with $\mathcal{E}_0 := \ni_0$ and $\mathcal{E}_1 := \mathsf{LB}_1 \xrightarrow{\langle \ni_0 \sigma, \mathsf{id}\rangle} \mathsf{LC}_0 \times_{\mathsf{LB}_0} \mathsf{LB}_1 \xrightarrow{\varepsilon_1} \mathsf{LC}_1$ which more explicitly can be computed as $\mathcal{E}_1^{-1} w = \lambda m.\ni_0^{-1} m^{-1} w$. By proposition 6.3 and lemma 6.4, the local homeomorphism σ_{LB} is induced by the $B_{\mathcal{J}}$-set $\Gamma\mathsf{LC}1$, but this just corresponds to the sheaf induced by the local homeomorphism σ_{LC}, so we must have $\sigma_{\mathsf{LC}} \cong \sigma_{\mathsf{LB}}$. The map $\mathcal{E}_1$ is the canonical map witnessing this isomorphism, up to a change of base along the isomorphism $\ni_0$. We leave it to the reader to verify functoriality.

A.21 Proof of theorem 6.8

It follows from proposition 6.7 that $\varepsilon_{\mathsf{LB}}$ is an isomorphism, since $\mathsf{LB}T$ is ample for any monad T. Hence adjunction 5.4 is idempotent, and propositions 6.5 and 6.7 characterize the fixpoints. For the finitary monad case, we know that $\mathcal{O} : \mathsf{Top} \to \mathsf{Loc}$ preserves pullbacks along local homeomorphisms, which means it preserves ample topological categories and their retrofunctors. From this, we get a functor $\mathcal{O} : \mathsf{AmpTopRetro} \to \mathsf{AmpLocRetro}$, for which the equivalence $\mathsf{HUMnd}_r \simeq \mathsf{AmpLocRetro}$, when restricted to finitary monads, factors through. This factorization $\mathsf{HUMnd}_\omega \to \mathsf{AmpTopRetro}$ is essentially surjective on

objects, because now taking the global sections monad on an ample topological category $\mathbb{C}$, the compactness of the base space $\mathbb{C}_0$ ensures this monad is the monad $\Gamma_\omega\mathbb{C}$ of *finitary* sections. Hence, we get an equivalence $\mathsf{HUMnd}_\omega \simeq \mathsf{AmpTopRetro}$.