Electronic Notes in Theoretical Computer Science 122 (2005) 247–296 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Inside Every Model of Abstract Stone Duality Lies an Arithmetic Universe

# Paul Taylor[1](#_bookmark4)

*Department of Computer Science, University of Manchester*

Abstract

The first paper published on *Abstract Stone Duality* showed that the *overt discrete* objects (those admitting ∃ and = internally) form a *pretopos*, *i.e.* a category with finite limits, stable disjoint co- products and stable effective quotients of equivalence relations. Using an N-indexed least fixed point

axiom, here we show that this full subcategory is an *arithmetic universe*, having a *free semilattice*

(“collection of *Kuratowski-ﬁnite* subsets”) and a *free monoid* (“collection of *lists*”) on any overt discrete object. Each finite subset is represented by its pair ( , ♦) of *modal operators*, although a tight correspondence with these depends on a stronger Scott-continuity axiom. Topologically, such subsets are both *compact and open* and also involve *proper open maps*. In applications of ASD this can eliminate lists in favour of a *continuation-passing interpretation*.

*Key words*: Abstract Stone Duality, pretopos, arithmetic universe, free semilattice, Kuratowski- finite, lists, modal logic, powerdomains, proper open maps.

Contents 6. Admissible implies modal

|  |  |  |  |
| --- | --- | --- | --- |
| 1. Introduction | [247](#_bookmark0) | 7. K as a functor | [273](#_bookmark53) |
| 2. Proofs and natural numbers | [252](#_bookmark7) | 8. Lists, heads and tails | [278](#_bookmark66) |
| 3. Finite subsets as modalities | [259](#_bookmark11) | 9. Recursion over lists | [282](#_bookmark72) |
| 4. Fixed point properties | [263](#_bookmark21) | 10. The free semilattice | [287](#_bookmark85) |
| 5. Stages in the construction | [267](#_bookmark33) | 11. Overt compact subspaces | [290](#_bookmark92) |

[271](#_bookmark45)

# Introduction

In Abstract Stone Duality the topology on a space *X* is treated as an exponen- tial Σ*X* with a *λ*-calculus rather than as a lattice with arbitrary joins. This has given accounts of the category of locally compact spaces, both over an

1 Correspondence by email only, please, to <[pt@cs.man.ac.uk](mailto:pt@cs.man.ac.uk)>

1571-0661 © 2005 Paul Taylor. Published by Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2004.06.059

elementary topos [[H](#_bookmark113)], and for a logic in which the maps N → N are precisely the provably total general recursive functions [[G](#_bookmark112)].

Remark 1.1 It is an important feature of ASD that its spaces do not have “underlying sets” of points — or even of open subspaces, as in locale theory. ASD is a *direct* axiomatisation of the category S of “spaces”, amongst which

the “discrete” ones serve in the role of “sets”. However, as we take the word

discrete to mean that there is an internal notion of equality, (=*X*) : *X* ×*X* → Σ (*i.e.* the diagonal *X* ⊂ *X* × *X* is open), we actually say *overt discrete*, meaning that there is also an “existential quantifier”, ∃*X* : Σ*X* → Σ.

Remark 1.2 Having postulated a notion of the category of “sets” in this roundabout fashion, *i.e.* as the full subcategory E of overt discrete types in a certain *λ*-calculus, we are faced with the challenge of showing that it has

enough of the usual features of set theory or categorical logic to warrant the

name, as none of these went in as ingredients. In fact, E is a topos if we assume the existence of “underlying sets”, *i.e.* a right adjoint to the inclusion E *‹*→ S [[H](#_bookmark113)], so it is important *not* to make such an assumption if we want to develop a computational axiomatisation of topology.

Giraud’s theorem, which characterised Grothendieck toposes in terms of the limits and (infinitary) colimits that they admit, suggested the first cate- gorical approximation to the “finitary” aspects of the category of sets: a *pre- topos* is a category with finite limits, stable disjoint coproducts and sta- ble effective quotients of equivalence relations. Then Andr´e Joyal introduced *arithmetic universes* to prove Go¨del’s incompleteness theorem in a cate-

gorical style: they have just enough structure to form the *free internal* thing

of the same kind. Specifically, an arithmetic universe is a pretopos with free internal monoids (List*X*), from which the free internal arithmetic universe can be obtained by means of generators and relations.

Actually, these structures are just what is taught in a first year “discrete math” course intended for computer scientists, *i.e.* one in which only the col- lection of *listable* subsets is considered, instead of a full set-theoretic powerset. We shall argue in future work (on the construction of S from E) that the other

substitutes for the powerset, *i.e.* the collections of *recursively enumerable* and

of *decidable* subsets of *X*, are the non-discrete spaces Σ*X* and 2*X*. If Σ*X* had an underlying set, this would be the usual powerset of *X*.

Remark 1.3 Unfortunately, despite their 30-year history, knowledge of arith- metic universes circulates literally by word of mouth. The only only refereed (or even obtainable) papers on them are those by Maria Emilia Maietti, who provided a Martin-L¨of-style type theory for them [[6](#_bookmark103)]. She claims that her

notion of arithmetic universe [[7](#_bookmark104)] is stronger than that used here. Indeed, some disagreement over detail is known by those who have worked with arithmetic universes — but they hardly help matters by failing to write any papers. How- ever, the best way to settle doubts over correct definitions in mathematics is to prove their equivalence with other structures that come from other intu- itions. The construction in this paper of an arithmetic universe from a model of ASD and the proposed converse will, I believe, not only serve this purpose, but also provide a more expressive calculus than Maietti’s, allowing domain- theoretic methods to be used to define structures within arithmetic universes more fluently.

Remark 1.4 In ASD the full subcategory of overt discrete objects is already known to be a pretopos [[C](#_bookmark106)], so the present work is concerned with the free monoid and the free semilattice (K*X*).

Note, however, that, as an existential question, this is redundant in the two cases on which we usually focus: in the classical models (the source of our

intuitions) E is a topos, whilst in the free one (the target of our computations) every object of E is a subquotient of N by an open partial equivalence relation; in both cases lists can be encoded in well known ways.

This paper is also arguably unnecessary from a topological point of view, in that the existence of the free monoid could reasonably be taken as another axiom, so our conclusion is merely that this axiom is redundant.

Nevertheless, ASD sets itself apart from other approaches to topology by having a *freestanding* (technically) elementary axiomatisation that does not rely on a pre-existing category of sets or spaces. The intention is to use this as a route to computation, so our *representation* of K*X* is of interest even when we already know abstractly that it exists.

Remark 1.5 The two semilattice structures that we use most often are ∧ and

∨ in powers of Σ, and it turns out that these are jointly faithful, indeed that K*X* is a subspace of Ω ≡ ΣΣ*X* × ΣΣ*X* , where each finite subset *l* is represented by the two *modal operators*,

[*l*] ≡ *λφ.* ∀*x* ∈ *l. φx* and ⟨*l*⟩ ≡ *λφ.* ∃*x* ∈ *l. φx.*

It is well known that modal logic is related to the three *powerdomains*, in one of which the inclusion order agrees with the intrinsic one, in another they are contravariant, whilst in the third inclusion inclusion involves both the intrinsic order and its opposite [[4](#_bookmark99),[9](#_bookmark107),[10](#_bookmark108)]. However, we only consider the convex powerdomain of an (overt) discrete space. The reason for this lack of ambition is that this paper forms part of the “bootstrapping” of the theory

of ASD: much more can be done using the whole thing, in particular [[G](#_bookmark112)]; the analogous structure in an Hausdorff space such as R is considered in [[I](#_bookmark114)]. Also,

we represent *elements* of the powerdomain by modal operators, whereas the

works cited use them to generate its *topology*, relying on the prior existence of the free semilattice.

Remark 1.6 Our translation of finite subsets into higher order *λ*-terms may be of computational value, like other examples of the *continuation passing style*. The theory of locally compact spaces is developed in [[G](#_bookmark112)] using bases of open and compact subspaces, and that paper concludes with a sketch of

how continuous functions (for example R → R) might be manipulated com-

putationally using relations. Unfortunately, at least as the theory is expressed there, such bases have to be indexed by (finitary) *lattices*, and heavy use is

made of the free distributive lattice generated by an overt discrete object. This may be constructed *via* the free semilattice, whose elements might in turn be represented as *lists*, but possibly at quite a heavy computational cost.

On the other hand, as functional programmers know very well, there are plenty of situations in which (nested) lists provide excellent data structures. For this, it would be absurd to take a diversion *via* logic, especially by means of the construction that we use in Sections [8](#_bookmark67)–[9](#_bookmark73). What this paper provides, therefore, is a *choice* between lists (as traditionally implemented) and a math- ematically isomorphic structure that encodes finite subsets using modal logic, *λ*-calculus and continuation-passing. Empirical study will be needed to make

this choice in particular applications.

Remark 1.7 Returning to type theory, the disagreement with Maietti regard- ing the definition of an arithmetic universe appears to concern *parametric* list types, which are not covered in this paper. Certainly these are needed, but there are ways of obtaining them without first developing dependent spaces

in general.

A family of overt discrete objects indexed by another such object is given, as in a topos, by *any* map *δ* : *X* → *N* , where *X*[*n*] ≡ *δ*−1(*n*). (We call *δ*

the “display” map [[8](#_bookmark105), Chapter VIII].) Then List*X*[*n*] is given by the display

*N* + *P* → *N* , where

*P*  ) *N*

v

List*X*

List*δ*)

List*N*

))

{|−}|

v

K*N*

provides the non-empty lists and the extra term *N* the empty ones.

Remark 1.8 Another connection between our construction and domain the- ory is that it uses the methods of that subject. Specifically, the ASD calculus provides for the introduction of subtypes of a given ambient type Ω, defined

by an endomorphism E of ΣΩ. In this paper we obtain that endomorphism as the least fixed point of an operator $ of yet higher type.

To do this, a fixed point axiom must be added to what [[C](#_bookmark106)] used to show that E is a pretopos. To construct K*X* and List*X* it is enough to assume

Axiom 1.9 The “linear countable” *fixed point axiom* is

Γ*, n* : N ▶ *φn* : Σ*U* Γ*, n* : N ▶ *φn* ≤ *φ*(*n* + 1) : Σ*U* Γ*, F* : Σ*U* → Σ*V* ▶ *F* (∃*n. φn*) = ∃*n. F* (*φn*) : Σ*V*

However, in order to obtain all of the topological results that we expect [[G](#_bookmark112)], and in particular to show that every pair ( *,* ♦) of operators satisfying the modal laws actually corresponds to a listable subset(Section [11](#_bookmark95)), the following stronger [2](#_bookmark6) assumption is needed. Despite the conceptual simplification that

it would bring throughout this paper, we are unable even to state it before proving our main result, as it involves either List*X* or K*X*.

Axiom 1.10 The *Scott-continuity axiom* is

Γ*, l* : K*X* ▶ *φl* : Σ*U* Γ*, l*1*, l*2 : K*X* ▶ *φl*1 ∨ *φl*2 ≤ *φl*1+*l*2

Γ*, l* : K*X* ▶ *αl* : Σ Γ ▶ *α* nil = T Γ*, l*1*, l*2 : K*X* ▶ *αl*1 +*l*2 = *αl*1 ∧ *αl*2

Γ*, F* : Σ*U* → Σ*V* ▶ *F* (∃*l. αl* ∧ *φl*) = ∃*l. αl* ∧ *Fφl.*

A family (*αl, φl*) satisfying the five premises is called a *directed diagram* . In the free model every overt discrete object such as K*X* is a subquotient of N; in this case general Scott-continuity may be derived from the fixed point axiom above. In other models, such as the classical ones, a Scott-continuity axiom is needed for each overt discrete object *X*.

Remark 1.11 This brings us back to the study of *general* models of ASD (for which the existential question about K*X* and List*X* is not trivial). Besides

2 The fixed point axiom says that F preserves *countable* directed joins, but in the classical models *uncountable* directed joins are also needed. Recall, for example, that in universal algebra the free functor for a theory of infinite arity κ preserves κ-filtered colimits of dia- grams of any size λ. The two versions of our axiom correspond very roughly to the roles of these two cardinal parameters in the classical case.

the free and classical ones, we may obtain other models by strengthening the axioms with more ways of forming types, terms or equations. For example,

the “underlying set” is a new *type* [[H](#_bookmark113)]. An oracle for termination would say that the (already definable) space *T*¯ of codes for non-terminating programs is overt, providing a new *term* ∃ ¯ : Σ*T*¯ → Σ.

*T*

The very meaning of the assertions that we made in the opening paragraph

depends on what we can prove about *equality* of terms. A morphism of the category is a class of terms that are provably equivalent according to a certain logic, whilst a partial morphism *f* : N *~* N is total by definition iff there is a

proof that Σ*f* satisfies a certain equation [[D](#_bookmark111)]. Since the definitions of *prime*

and *nucleus* [[A](#_bookmark109),[B](#_bookmark110)] depend on equations, if the logic proves more of them then it also defines more terms and (sub)types.

What counts as a legitimate proof (in particular of the equality of two terms) is an issue that we have to consider more carefully in this paper than has been done hitherto in the ASD literature, because this is the first *serious* use that we have made of recursion and induction over N. That is the subject of the next section.

Section [3](#_bookmark12) gives the intuition and notation for the construction of K*X* and its modal logic, which are developed in Sections [4](#_bookmark24)–[6](#_bookmark47). Section [7](#_bookmark54) considers the properties of K as a functor. The universal property of K*X* as the free semilattice is, however, only proved in Section [10](#_bookmark86), being derived from that for List*X*, which we construct in Sections [8](#_bookmark67)–[9](#_bookmark73). Section [11](#_bookmark95) reconsiders the sense in which K*X* is a finite powerset.

# Proofs and natural numbers in ASD

Before we begin the construction of the free monoid and semilattice, we have to consider the form of proofs in the *λ*-calculus for ASD more carefully than has been done in previous papers. Nevertheless, our purpose is a domain-

theoretic construction, not a proof-theoretic analysis of ASD, so we shall not give the complete set of rules. See in particular [[B](#_bookmark110), §8] for a summary of the *λ*-calculus for {*X* | *E*} that handles the monadic property.

The main issue to be considered is induction over N, the point being that the definition of the natural numbers object is not adequate as it is usually given. It is well known that an object Γ of *parameters* has to be added explicitly to the definition when we work in a category that is not *cartesian closed*. Similarly, *equational hypotheses* have to be considered explicitly when the category does not have all *equalisers*.

Axiom 2.1 The *λ*-calculus for ASD consists of types, terms and equations.

Its judgements assert that

* + types such as 0, 1, N, Σ, *X* × *Y* , Σ*X* and {*X* | *E*} are well formed,
  + terms are well formed and of particular types, or
  + equations hold between terms.

Such judgements about terms and equations are made in certain *contexts*,

*i.e.* on the assumption that their free variables have certain types.

However, even though the definition of the subtype {*X* | *E*} involves the term *E*, we insist (as the theory is currently formulated) that it be formed in the empty (global) context, *i.e.* without free variables. Even the pure syntax of dependent type theories is very complicated, and becomes more

so in a semantic situation, where we have to choose a class of display maps [[8](#_bookmark105), Chapter VIII]. Of course, we must also perform the semantic calculations, which form the main task of this paper. Families of overt and compact objects

are encoded as open and proper maps respectively [[C](#_bookmark106), §7].

Judgements are therefore of the four forms

▶ *X* type*,* Γ ▶ *a* : *X,* Γ ▶ *a* = *b* : *X* and Γ ▶ *α* ≤ *β* : *X.*

The last of these arises from the lattice structure on types of the form Σ*U* , where (*α* ≤ *β*) means *α* = (*α* ∧ *β*) or *β* = (*α* ∨ *β*). The order can be extended to other objects, but we shall not need that.

In this paper we find that *equations* (and inequalities) *are also needed as assumptions*. In other words, the context Γ may include equations and in- equalities between terms, as well as a list of typed variables. Any provable judgement attests to the validity of a certain *fragment* of reasoning from its hypotheses to its conclusions, so, as we want to form bigger arguments by concatenating such fragments (*i.e.* by means of a cut rule), all forms of asser-

tion that are allowed as conclusions should also be allowed as hypotheses. In particular this is needed for induction over N, which internalises the process of concatenation. (Adding such hypotheses to Martin-Lo¨f type theory leads to undecidability [[2](#_bookmark100), §3.2], but the ASD calculus is Turing-complete by design

anyway.)

Axiom 2.2 Terms of type Σ may be seen as predicates in coherent logic:

1. T and ⊥;
2. *α* ∧ *β* and *α* ∨ *β*, where *α, β* : Σ;
3. *φa*, where *φ* : Σ*X* and *a* : *X*;
4. (*a* =*X b*), where *a, b* : *X*, and *X* is discrete (such as N but not R or 2N);
5. (*a* /=*X b*), where *X* is a Hausdorff type (such as N, R or 2N);
6. E*Xφ* ≡ E*x. φx*, where *φ* : Σ*X* and *X* is an overt type (N, R, 2N);
7. 6*Xφ* ≡ 6*x. φx*, where *φ* : Σ*X* and *X* is a compact type (such as [0*,* 1] ⊂ R

or 2N but not N).

We shall find that a type *X* has all four properties (d–g) iff it is finite, whilst it is *Kuratowski* -finite iff it has properties (d,f,g), that notion being the main subject of this paper. Terms of type *φ* : Σ*X* are formed with a *λ*-calculus. As usual, the introduction of *λx*, E*x* or 6*x* discharges the variable from the context, which must therefore contain no equational assumption in which *x* is

free.

Remark 2.3 Notice that this logic does not include implication. Indeed, every term is monotone considered as a function of its free variables of type Σ*U* . But a strictly limited form of implication is allowed in that assumptions

and conclusions of judgements may be of the form *α* ≤ *β* : Σ*U* , so judgements may be of the form

*... ,α* ≤ *β* : Σ*U ,...* ▶ *γ* ≤ *δ* : Σ*V ,*

which means (6*u. αu* ⇒ *βu*) ⇒ (6*v. γv* ⇒ *δv*) in traditional notation.

When we need to make an assertion *α* that is a term of type Σ, we shall

often follow mathematical idiom by saying simply “*α*” alone; by this we mean the judgement Γ ▶ *α* = T : Σ, where the context Γ has been established implicitly in the argument.

Of course, any equation or inequality that is an assumption may be used directly as the conclusion of an “identity” or “axiom” judgement, or may be discharged by a “cut” rule. They are also discharged by logical rules that we now give.

Axiom 2.4 Even though we cannot define a term *α* ⇒ *β* of type Σ, the usual rule for it does introduce an inequality, in both an intuitionistic way and an

apparently classical one:

Γ*, α* = T : Σ ▶ *β* = T : Σ

Γ ▶ *α* ≤ *β* : Σ

Γ*, α* = ⊥ : Σ ▶ *β* = ⊥ : Σ Γ ▶ *α* ≥ *β* : Σ

The intuitionistic rule is equivalent to the *Euclidean principle*,

*F* : ΣΣ*, α* : Σ ▶ *α* Λ *Fα* = *α* Λ *F* T*,*

and the other rule to its lattice dual. Writing *α* and ч*β* for *α* = T and *β* = ⊥, together they yield rules similar to those for negation in Gerhard Gentzen’s

classical sequent calculus.

They and monotonicity are together equivalent to the *Phoa principle*,

*F* : ΣΣ*, α* : Σ ▶ *Fα* = *F* ⊥∨ *α* Λ *F* T*.*

Note that these things are valid in intuitionistic locale theory [[C](#_bookmark106), Section 5]. The Euclidean principle is equivalent (in the context of the monadicity as-

sumption) to saying that T : Σ classifies a certain class of monos in S, which we call *open*, whilst its dual says that ⊥ : Σ classifies *closed inclusions*.

An example of the first principle that’s obvious in the abstract but may look strange when it’s used in practice is the rule (where *X* and *Y* are discrete)

Γ*, a* = *b* : *X* ▶ *c* = *d* : *Y*

Γ ▶ (*a* =*X b*) ≤ (*c* =*Y d*) : Σ

Turning to recursion and induction, we have to reconsider the definition

of N in a category without equalisers, such as that of locally compact spaces. See also [[1](#_bookmark98)] and [[5](#_bookmark102), §II.4].

Axiom 2.5 The *recursion scheme* introduces *terms* of any type *X*, depen- dent on *n* : N:

Γ ▶ *z* : *X* Γ*, n* : N*, x* : *X* ▶ *s*(*n, x*) : *X*

Γ*, n* : N ▶ rec(*n, z, s*) : *X* Its meaning is given by the *β*-rules

Γ ▶ rec(0*, z, s*) = *z* : *X*

Γ*, n* : N ▶ rec(*n* + 1*, z, s*) = *s* *n,* rec(*n, z, s*) *.*

This is called *recursion at type X*, the point being that as the class of types at which the recursion scheme is asserted grows, so considerably does the power of the logic.

The corresponding diagrammatic property is

Γ 0 ) Γ × N (id × succ Γ × N

.

.

.

. rec

.

.

.

.

.

.

. (id*,* rec)

.

*z* . .

.v *s* .v

*X* ( Γ × N × *X.*

If the category is cartesian closed then this diagram may be rewritten with

N × *X*Γ in place of *X*, the object Γ itself being removed from the top line.

Symbolically, this means that the parameters can be embedded in both the data and the new term by means of *λ*-abstraction. If, as in ASD, the category (has finite products but) is not cartesian closed then the object Γ of parameters is needed.

Axiom 2.6 The *equational induction scheme* for Γ*, n* : N ▶ *an, bn* : *X*: Γ ▶ *a*0 = *b*0 : *X* Γ*, n* : N*, an* = *bn* : *X* ▶ *an*+1 = *bn*+1 : *X*

Γ*, n* : N ▶ *an* = *bn* : *X*

Notice that this says nothing about where *an* and *bn* themselves *came* from. If they had been defined by the same base and step data on *X*, an *η*-rule or uniqueness hypothesis in the recursion scheme above would have made them equal. However, the scheme is more general than that: the term *an* need no t

arise *directly* from recursion, but may perhaps be of the form *f*

rec(*n, z, s*)

where *f* is some function, whilst *bn* is some unrelated *f* ' rec(*n, z*'*, s*') .

Remark 2.7 Just as the generalisation with parameters was redundant in a cartesian closed category, so the equational induction scheme can be derived from the usual universal property of N when the category has equalisers:

*E* ≡ {(*γ, n*) | *an*(*γ*) = *bn*(*γ*) : *X*}) )

*an*)

Γ × N ) *X.*

*bn*

The base case says that (id*,* 0) : Γ → Γ × N factors through *E*, whilst the induction step says that id × succ restricts to a map *E* → *E*. Then Axiom [2.5](#_bookmark9) provides

rec*E* : Γ × N → *E.*

This is inverse to the inclusion, one equation being given by uniqueness of

recN : Γ × N → Γ × N

and the other by the fact that *E* > Γ × N is mono.

The equational induction scheme (and its equivalent version for inequalities *αn* ≤ *βn*) will be used in Propositions [4.4](#_bookmark26), [4.12](#_bookmark34), [5.9](#_bookmark42) and [7.10](#_bookmark62), and we shall deduce its analogue for lists and finite subsets in Propositions [9.7](#_bookmark80) and [10.11](#_bookmark93).

It ought to have been stated as [[A](#_bookmark109), Remark 2.5], since it was actually used in [[A](#_bookmark109), Lemmas 8.9 and 9.6] and [[B](#_bookmark110), Lemma 8.14].

Since the proof given for [[A](#_bookmark109), 9.6] was sketchy and contained other errors, we give the correct version here by way of an example of equational induction.

It turns recursion at type N into recursion at type ΣN, as part of the process of bringing descriptions (the *m.* ) to the outside of a term.

Lemma 2.8 Γ ▶ *rn* ≡ rec *n, z, λn*'*u. s*(*n*'*, u*) = the *m. ρ*(*n, m*), where

*ζ* ≡ (*λm. m* = *z*)*,*

*σ*(*n, φ*) ≡ *λm.* E*m*'*. φm*' Λ *m* = *s*(*n, m*')

*ρ*(*n, m*) ≡ rec(*n, ζ, σ*)*m.*

Proof The base case (in context Γ) is

*λm. ρ*(0*, m*) ≡ *λm.* rec(0*,ζ, σ*)*m* = *λm. ζm β*-rule

≡ *λm.* (*m* = *z*) def *ζ*

= *λm.* (*m* = *r*0) *β*-rule

The induction step, with the hypothesis *λm*'*. ρ*(*n, m*') = *λm*'*.* (*m* = *rn*'), is

*λm. ρ*(*n* + 1*, m*) ≡ *λm.* rec(*n* + 1*,ζ, σ*)*m*

= *λm. σ* *n,* rec(*n, ζ, σ*) *m β*-rule

= *λm.* E*m*'*.* rec(*n, ζ, σ*)*m*' Λ *m* = *s*(*n, m*') def *σ*

≡ *λm.* E*m*'*. ρ*(*n, m*') Λ *m* = *s*(*n, m*')

= *λm.* E*m*'*.* (*m*' = *rn*) Λ *m* = *s*(*n, m*') hypothesis

= *λm.* *m* = *s*(*n, rn*) equality rules

= *λm.* *m* = *r*(*n* + 1) *β*-rule So Γ*, n* : N ▶ *λm. ρ*(*n, m*) = *λm.* (*m* = *rn*) : Σ by Axiom [2.6](#_bookmark10).

Remark 2.9 Recall from [[B](#_bookmark110), Section 8] that the rules for the introduction of focus and admit terms have equational premises. This now means that there are such terms that are only defined in certain contexts that contain particular equational assumptions.

Recall, however, from *op. cit.* that, for any term Γ ▶ *α* : Σ, there is another term Γ ▶ *α*¯ : Σ not involving focus or admit, with the property that Γ ▶ *α* = *α*¯ : Σ. (In fact, *α*¯ is obtained, essentially, by *erasing* admit.) Now, as *α*¯ does not contain focus or admit, no equational hypotheses are ever used in its formation, and so it is defined in the context Γ¯ without them.

Corollary 2.10 Σ is injective in the category of contexts (possibly involving equations) and substitutions.

Remark 2.11 How might such equational hypotheses be interpreted in a category S that does not necessarily have all equalisers? The obvious way is that, for *f*1*, f*2 : *X* → *A* and *g*1*, g*2 : *X* → *B* in S, the statement

*x* : *X, f*1*x* = *f*2 : *A* ▶ *g*1*x* = *g*2*x* : *B*

means that

for every *a* : Γ → *X* in S*,* if (*f*1 · *h*) = (*f*2 · *h*) then (*g*1 · *h*) = (*g*2 · *h*)*.*

The restricted *λ*-calculus was formally extended with (abstract) sobriety [[A](#_bookmark109)] and monadicity [[B](#_bookmark110)]. This interpretation of equational hypotheses could be used to make another similar formal extension.

However, the result of such an extension would be an account of *spatial* locales (equivalently, sober spaces — in the traditional sense rather than that of [[A](#_bookmark109)]), not general ones. This is because the generality of the test object Γ

is spurious: locally compact locales have enough points (classically, at least), and so this formula only uses the global points of Γ to test the equations. Equalisers (and, in fact, products) of locales and sober spaces are not the same [[3](#_bookmark101), §II 2.13].

Remark 2.12 The category whose objects are contexts with equations has finite products, but it no longer has the exponentials Σ(−) that originally mo- tivated ASD — these are only defined on the subcategory of contexts without equations. The problem is inescapable since, as we have just noted, such hy- potheses have already been used, albeit inadvertently. This means that the theory really captures, not the category of locally compact spaces on its own, but that category embedded in either the category of locales or or spatial locales.

Nevertheless, the way forward is not to rewrite what has been done in this hybrid fashion, but to study the (*substantially*) larger structure that in- cludes both equalisers and exponentials. Preliminary investigations of this structure, relating not only to the whole of the category of locales but also to

cartesian closed extensions, may be found in [[H](#_bookmark113)]. However, this structure not only contains spaces of a much more general kind, but also captures a much stronger logic of (implications amongst) equations. This means that, in order to nail down the precise equivalence with arithmetic universes (Remark [1.3](#_bookmark1)), the proposed converse construction will probably be of a category analogous to spatial locales.

But for the purposes of the present paper, the equational hypotheses are simply a device for managing proofs.

# Finite subsets as modal operators

Throughout, let *X* be an overt discrete space. Its typical values will be called *x*, *y*, *etc.* and its predicates *φ*, *ψ*, *etc*. We shall construct an object K*X* and then show that it is the free semilattice on *X*. Although K*X* is intended to be the space of Kuratowski-finite *subsets* of *X*, we shall use *list* notation for

it, writing *l* : K*X* for a typical value. So nil *,* {|*x*}| : K*X* denote the empty and

singleton subsets, + is union and *x* :: *l* = {|*x*|} + *l*. Note that we also use

{*x*} for *λy.* (*x* =*X y*) : Σ*X* . We shall actually need to switch back and forth between K*X* and List*X*, and between the “mathematical” notion of monoid

based on the + operation and the “computer science” notion based on ::.

Remark 3.1 K*X* will be constructed as a Σ-split subspace of Ω ≡ ΣΣ*X* ×ΣΣ*X* .

The typical values of Ω will be called *L* = (*N, P* ), where the letters stand

for *necessity* and *possibility* , as the central idea is to represent a subset *l* : K*X* by means of its *modal operators* [*l*]*,* ⟨*l*⟩ : ΣΣ*X* , which also vary *negatively* and *positively* with respect to the inclusion of lists. Along with *P* : ΣΣ*X* we shall also make frequent use of *π* ≡ *λx. P* {*x*} : Σ*X*.

Remark 3.2 If we already had K*X* and List*X* at our disposal, we would define

[*l*] ≡ *λφ.* 6*x* ∈ *l. φx* and ⟨*l*⟩ ≡ *λφ.* E*x* ∈ *l. φx,*

and the *membership predicate* (*x* ∈ *l*) ≡ *πlx* = *P* {*x*}, where *P* ≡ ⟨*l*⟩. Then

|  |  |  |  |
| --- | --- | --- | --- |
| [ nil ] = *λφ.* T | ⟨ nil ⟩ = *λφ.* ⊥ | *π* nil | = *λy.* ⊥ |
| [*x* :: *l*] = *φx* Λ [*l*]*φ* | ⟨*x* :: *l*⟩ = *φx* ∨ ⟨*l*⟩*φ* | *πx*::*l* | = *λy.* (*x* = *y*) ∨ *πly* |
| [*l*1 + *l*2] = [*l*1] Λ [*l*2] | ⟨*l*1 + *l*2⟩ = ⟨*l*1⟩∨ ⟨*l*2⟩ | *πl*1 +*l*2 | = *πl*1 ∨ *πl*2 *,* |

so that [{|*x*|}] = ⟨{|*x*|}⟩ = *ηx* = *λφ. φx* and *π*{|*x*}| = {*x*}. Also

(*l*1 ⊂ *l*2) ≡ 6*x* ∈ *l*1*.* E*y* ∈ *l*2*.* (*x* =*X y*) ≡ [*l*1](*λx.* ⟨*l*2⟩{*x*}) ≡ [*l*1]*π*2

and (*l*1 = *l*2) ≡ (*l*1 ⊂ *l*2) Λ (*l*2 ⊂ *l*1), which are values of type Σ. Finally, the “Curried” operations *φ* ≡ *λl.* [*l*]*φ* and ♦ *φ* ≡ *λl.* ⟨*l*⟩*φ* generate the topology on the powerdomains (Remark [1.5](#_bookmark2)), but are not very useful here.

Notation 3.3 As we want to construct K*X* as a subspace of Ω, we need to extend this notation to *L* ≡ (*N, P* ) : Ω ≡ ΣΣ*X* ×ΣΣ*X* . For the modal notation itself, we simply use [ ] and ⟨ ⟩ for the product projections, so *φL* ≡ [*L*]*φ* ≡

*Nφ* and ♦ *φL* ≡ ⟨*L*⟩*φ* ≡ *Pφ*. Then we define

nil ≡ (T*,* ⊥) : Ω

*x* :: (*N, P* ) ≡ (*x* :: *N, x* :: *P* ) ≡ (*λφ. Nφ* Λ *φx, λφ. Pφ* ∨ *φx*) (*N*1*, P*1)+ (*N*2*, P*2) ≡ (*N*1 Λ *N*2*, P*1 ∨ *P*2)

≡ (*λφ. N*1*φ* Λ *N*2*φ, λφ. P*1*φ* ∨ *P*2*φ*)

*x* ∈ *L* ≡ ⟨*L*⟩{*x*} ≡ *P* {*x*} ≡ *πx*

*L*1 ⊂ *L*2 ≡ [*L*1](*λx.* ⟨*L*2⟩{*x*}) ≡ *N*1(*λx. P*2{*x*}) ≡ *N*1*π*2 *L*1 ∼ *L*2 ≡ (*L*1 ⊂ *L*2) Λ (*L*2 ⊂ *L*1)

*L* ∼ nil = [*L*]⊥ ≡ *N* ⊥

*L* /∼ nil = ⟨*L*⟩T ≡ *P* T*.*

Clearly + is an associative, commutative, idempotent binary operation on Ω, and nil is a unit for it, with nil ⊂ *L*; we shall derive their other algebraic properties of + and ⊂ shortly. Notice, however, that for *P* they behave like

∨ and ≤, but for *N* they are like Λ and ≥. This means that they are not

*intrinsic* structure on Ω (as ≤, ∨ and Λ are in Σ), but *imposed* on it by specifying certain maps.

The Λ/∨-ambiguity in the :: notation will always be resolved by the con- text, but we shall not risk confusion by further overloading of the important

sign for equality.

Plainly not every pair (*N, P* ) will arise as ([*l*]*,* ⟨*l*⟩) from a finite subset.

Definition 3.4 We say that the pair Γ ▶ (*N, P* ) : Ω is *modal* if *N* and *P*

satisfy

*N* T = T *P* ⊥ = ⊥

*N* (*φ* Λ *ψ*) = *Nφ* Λ *Nψ P* (*φ* ∨ *ψ*) = *Pφ* ∨ *Pψ N* (*φ* ∨ *ψ*) ≤ *Nφ* ∨ *Pψ P* (*φ* Λ *ψ*) ≥ *Pφ* Λ *Nψ*

*N* (*λx. P* {*x*}) = T *Pφ* = E*x. φx* Λ *P* {*x*}*.*

The last law, which recovers *P* from *π* ≡ *λx. P* {*x*}, implies that *P* preserves

⊥, ∨ and E. Recalling the classical connection between finiteness and Scott

continuity, it is perhaps not surprising that Axiom [1.10](#_bookmark5) provides converses. If *P* is Scott continuous and preserves ⊥ and ∨ then it also preserves E and satisfies the last law. From Scott-continuity of *N* , Theorem [11.7](#_bookmark96) shows that

the modal laws exactly characterise Kuratowski-finite subsets.

In view of the number of things to be checked, we shall omit most of the proofs that pairs (*N, P* ) are modal, recommending them as exercises. Note that the sixth and fifth laws usually require the Euclidean principle and its

dual respectively.

When the ambient space is Hausdorff instead of discrete (*e.g.* R), the seventh law is replaced by *P* (*λx. N* {*x*}), where {*x*} ≡ (*λy. x* /= *y*) [[I](#_bookmark114)].

Lemma 3.5 Modal operators satisfy the Frobenius law and its dual:

*φ* : Σ*X, σ* : Σ ▶ *P* (*σ* Λ *φ*) = *σ* Λ *Pφ* and *N* (*σ* ∨ *φ*) = *σ* ∨ *Nφ.*

Proof By the Phoa principle (Axiom [2.4](#_bookmark8)), since *P* ⊥ = ⊥ and *N* T = T.

Lemma 3.6 If (*N, P* ) is modal then *P* {*x*}Λ *Nφ* ≤ *φx*.

Proof

*P* {*x*}Λ *Nφ* ≤ *P* ({*x*}Λ *φ*)

≡ *P* *λy.* (*x* = *y*) Λ *φy*

= E*y. P* {*y*}Λ (*x* = *y*) Λ *φy*

= *P* {*x*}Λ *φx* ≤ *φx*

Using these laws we can already recover the algebraic structure of K*X*.

Proposition 3.7 Definable finite subsets give rise to modal pairs because

1. nil ≡ (T*,* ⊥) is modal;
2. *x* : *X* ▶ {|*x*|} ≡ (*λφ. φx, λφ. φx*) is modal;
3. if Γ ▶ (*N*1*, P*1)*,* (*N*2*, P*2) : Ω are modal then so is (*N*1*, P*1)+ (*N*2*, P*2).

A similar study of the intersection operation raises topological questions.

Proposition 3.8 There is a greatest modal (*N, P* ) iff *X* is *compact* , in which case *N* = 6*X* , *P* = E*X* and *π* = T.

Proof Suppose that there is a bound for all singletons:

*λx.* {|*x*|} ⊂ (*N, P* ) = *λx.* [{|*x*|}](*λy. P* {*y*}) = *λx. P* {*x*} = *π.*

Hence if (*N, P* ) is the greatest modal pair then *π* = T. In this case,

*P* = *λφ.* E*x. πx* Λ *φx* = *λφ.* E*x. φx* = E*X.*

Also, *x* : *X, φ* : Σ*X* ▶ *Nφ* = *Nφ* Λ *P* {*x*} = *φx,*

so Σ!*X σ* ≡ *λx. σ* ≤ *φ* iff *σ* ≤ *N* (*λx. σ*) ≤ *Nφ* ≤ *φx*, which means that Σ!*X* E *N* , *i.e. N* = 6*X* and *X* is compact.

Now *N* (*λx.* E*X*{*x*}) = *N* (*λx.* E*y. x* = *y*) = *N* T = T so if *X* is compact then (*N, P* ) ⊂ (6*X,* E*X* ). This pair is modal because, in particular,

E*x.* (*φx* Λ *ψx*) ≥ E*x. φx* Λ (6*y. ψy*) ≥ (E*x. φx*) Λ (6*y. ψy*)

6*x.* (*φx* V *ψx*) ≤ 6*x. φx* V (E*y. ψy*) ≤ (6*x. φx*) V (E*y. ψy*)

by the Frobenius law and its dual.

Proposition 3.9 If (=*X*) is decidable (in which case we say that *X* is *Haus-* *dorff* as well as discrete) and (*N*1*, P*1) and (*N*2*, P*2) are modal then so is

(*N, P* ), where *Nφ* = *N* *λx. φx* V *N* (*λy. x* /= *y*)

1

2

*Pφ* = *P*1 *λx. φx* Λ *P*2(*λy. x* = *y*)

= E*x. P*1{*x*}Λ *P*2{*x*}Λ *φx,*

and this is the meet with respect to ⊂. Decidability is necessary because,

if {|*x*|} and {|*y*}| have a meet (*N, P* ) that is modal, then *N* ⊥ = (*x* /= *y*), so

(*x* =*X y*) is decidable. Intersection of decidable lists can also be defined by a

“filtering” program — after we have proved that recursion is valid.

The modal laws are also enough to ensure that ~ provides the internal equality for K*X*.

Proposition 3.10 Let Γ ▶ *L*1*, L*2 : Ω be modal. Then these are equivalent: (*L*1 ⊂ *L*2) = T ⟨*L*1⟩ ≤ ⟨*L*2⟩ *π*1 ≤ *π*2 [*L*2] ≤ [*L*1]*.*

Proof By the 8th modal law, *P*1 ≤ *P*2 iff *π*1 ≤ *π*2. Recall that

(*L*1 ⊂ *L*2) ≡ [*L*1](*λx.* ⟨*L*2⟩{*x*}) ≡ *N*1(*λx. P*2{*x*}) ≡ *N*1*π*2*.*

By the 7th modal law this is implied by *π*1 ≤ *π*2 since T = *N*1*π*1 ≤ *N*1*π*2 ≡

(*L*1 ⊂ *L*2)*,* or by *N*2 ≤ *N*1 since T = *N*2*π*2 ≤ *N*1*π*2 ≡ (*L*1 ⊂ *L*2)*.*

Conversely, from *L*1 ⊂ *L*2, we deduce successively that

Γ*, φ* : Σ*X, N*2*φ* = T ▶ *π*2 ≤ *φ* Lemma [3.6](#_bookmark16) Γ*, N*1*π*2 = T ▶ *π*1 ≤ *π*2 similarly

Γ*, N*1*π*2 = T ▶ *P*1 ≤ *P*2 8th modal law Γ*, N*1*π*2 = T*, φ* : Σ*X, N*2*φ* = T ▶ *π*1 ≤ *π*2 ≤ *φ*

Γ*, N*1*π*2 = T*, φ* : Σ*X, N*2*φ* = T ▶ *N*1*π*1 ≤ *N*1*φ* monotonicity Γ*, N*1*π*2 = T*, φ* : Σ*X, N*2*φ* = T ▶ T ≤ *N*1*φ* 7th modal law

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Γ*,* | *N*1*π*2 = T*,* | *φ* : Σ*X* | ▶ *N*2*φ* ≤ *N*1*φ* | Axiom [2.4](#_bookmark8) |
| Γ*,* | *N*1*π*2 = T |  | ▶ *N*2 ≤ *N*1 |  |

Corollary 3.11 Amongst modal pairs, ~ provides the internal equality rela- tion, whilst ⊂ is the *imposed* partial order for which the imposed semilattice structure + is the join.

Proof The relation ⊂ is reflexive and transitive because the equivalent con- ditions on [*L*] or ⟨*L*⟩ are. We have Γ ▶ (*L*1 ~ *L*2) ≡ (*L*1 ⊂ *L*2) Λ (*L*2 ⊂ *L*1) = T iff Γ ▶ [*L*1] ≥ [*L*2], [*L*1] ≤ [*L*2], ⟨*L*1⟩ ≥ ⟨*L*2⟩ and ⟨*L*1⟩ ≤ ⟨*L*2⟩, which happens iff Γ ▶ *L*1 = *L*2 : Ω. As for the relationship with +,

(*L*1 ⊂ *L*2) = T E▶ ⟨*L*1⟩ ≤ ⟨*L*2⟩*,* [*L*1] ≥ [*L*2]

E▶ ⟨*L*2⟩ = ⟨*L*1⟩V ⟨*L*2⟩*,* [*L*2] = [*L*1] Λ [*L*2]

E▶ *L*2 = *L*1 + *L*2

Proposition 3.12 If (*N, P* ) are modal then (*N, P* ) ~ nil is decidable:

(*N, P* ) ~ nil = *N* ⊥ and (*N, P* ) /~ nil = *P* T = E*x. x* ∈ *L.*

Proof From the modal laws, *N* ⊥V *P* T ≥ *N* (⊥V T) = *N* T = T and *N* ⊥Λ *P* T ≤ *P* (⊥Λ T) = *P* ⊥ = ⊥. On the other hand, (*N, P* ) ~ nil = *N* ⊥ by definition, whilst *P* T = *P* (E*x.* {*x*}*x*) = E*x. P* {*x*} = E*x. x* ∈ *L*.

Corollary 3.13 K0 = 1 and K1 = 2.

Proof Ω = ΣΣ0 × ΣΣ0 ~= Σ × Σ and Ω = ΣΣ1 × ΣΣ1 ~= ΣΣ × ΣΣ.

0 1

By the Phoa principle and the constraints *N* T = T, *P* ⊥ = ⊥, *N* ⊥Λ *P* T = ⊥

and *N* ⊥V *P* T = T, we have (*N, P* ) ~ (T*,* ⊥) V (*N, P* ) ~ (id*,* id) .

Although we have seen that the modal laws characterise the *algebraic* struc- ture of K*X*, we still have to show that it is

1. a Σ-split subspace of Ω, as modal laws just define an equaliser;
2. the free semilattice on *X*, with induction and recursion, and
3. overt, making the quantifier E*l* : K*X* in Axiom [1.10](#_bookmark5) legitimate. We shall do this by developing another characterisation of K*X*.

# Fixed point properties

Since it is the purpose of the paper to *deﬁne* the space of Kuratowski-finite subsets, we have to eliminate them from the notation in the previous section. We illustrate this first with the extension of the *existential quantifier* on

E : ΣK*X* → Σ to an operator E : ΣΩ → Σ.

Remark 4.1 Transforming E*l. θl* into *θ* nil V E*x.* E*l. θ*(*x* :: *l*), we have

EK*Xθ* ≡ E*l. θl* = *θ* nil V E*x.* E*l. θ*(*x* :: *l*)

= *θ* nil V E*x.* EK*X* *λl. θ*(*x* :: *l*)

≡ *θ* nil V E*x.* EK*X* (S*xθ*)*.*

We can therefore define E as the least fixed point of E = *λ*Θ*.* Θ nil V E*x.* E(S*x*Θ)*,* where we write Θ for a typical value of type ΣΩ or predicate on Ω. Notice that unwinding this fixed point equation reveals the list representation

of subsets that we had managed to conceal behind the semilattice structure in the previous section.

Notation 4.2 The *shift operator* S : *X* × ΣΩ → ΣΩ is defined by

S*x*Θ(*N, P* ) ≡ Θ(*x* :: *N, x* :: *P* ) ≡ Θ(*λφ. Nφ* Λ *φx, λφ. Pφ* V *φx*)*,*

and the *exception operator* S : *X* × ΣΣ*X* → ΣΣ*X* by

S*xNφ* ≡ *N* (*λy. x* = *y* V *φy*)*.*

Remark 4.3 Similarly, to prove that the embedding *i* : K*X* > Ω is Σ-split, we must also show how to extend *any* predicate *θ* from K*X* to Ω, by means of a morphism *I* : ΣK*X* → ΣΩ. The composite E ≡ *I* · Σ*i* is called a *nucleus* (Definition [5.1](#_bookmark35)), *starting* from which, [[B](#_bookmark110)] shows how to *deﬁne* K*X* formally as

a subspace of Ω. Like E, E will be defined by a fixed point equation, the idea being that

*IθL* ≡ E*l.* (*L* ~ *l*) Λ *θl,* so EΘ*L* ≡ E*l.* (*L* ~ *l*) Λ Θ([*l*]*,* ⟨*l*⟩)*.*

We can expand this as before, since (*N, P* ) ~ *l* = *Nπl* Λ [*l*]*πP* :

EΘ(*N, P* ) = E*l. Nπl* Λ [*l*]*πP* Λ Θ([*l*]*,* ⟨*l*⟩)

= *Nπ* nil Λ [ nil ]*πP* Λ Θ nil

V E*x.* E*l. Nπx*::*l* Λ [*x* :: *l*]*πP* Λ Θ([*x* :: *l*]*,* ⟨*x* :: *l*⟩)

= (*N* ⊥ΛT Λ Θ nil )

V E*x.* E*l.* S*xNπl* Λ *P* {*x*}Λ [*l*]*πP* Λ S*x*Θ([*l*]*,* ⟨*l*⟩)

= (*N* ⊥Λ Θ nil ) V E*x. P* {*x*}Λ E(S*x*Θ)(S*xN, P* )*.*

The reasoning that has led up to this fixed point equation depends on the prior existence of lists, so we have to start again from this formula as our “guess” for the definition of the nucleus E. First let us recall the fixed

point property itself, which follows from Axiom [1.9](#_bookmark3), but also uses equational induction (Axiom [2.6](#_bookmark10)).

Proposition 4.4 Let *A* = Σ*U* and Γ ▶ *F* : *AA*. Then Γ ▶ *α* ≡ E*n. Fn*⊥ : *A* satisfies Γ ▶ *Fα* = *α*. Moreover, if Γ ▶ *β* : *A* is a *pre-fixed point* , that is, Γ ▶ *Fβ* ≤ *β*, then Γ ▶ *α* ≤ *β*.

Definition 4.5 Let E∞ and E∞ be the least fixed points of their respective equations above. In the next section we shall prove that E∞ is a nucleus on Ω, also satisfying

▶ E∞ ≤ id and *x* : *X,* Θ : ΣΩ ▶ E∞(S*x*Θ) ≤ S*x*($E∞Θ)*,*

so we define K*X* ≡ {Ω | E∞}. We call Γ ▶ *L* : Ω *admissible* if E∞ admits it: Γ*,* Θ : ΣΩ ▶ E∞Θ*L* = Θ*L.*

In Section [6](#_bookmark47) we shall show that all *admissible L* are *modal*, so we have the benefit of the algebraic structure that we described in the previous section.

Lemma 4.6 If Θ ≤ *λL. σ* then E∞Θ ≤ *σ*.

Proof First, E0Θ = Θ nil ≤ *σ*. Now, if E ≤ *λ*Θ*.σ* then E(S*x*Θ) ≤ *σ*, so ΦE ≡ *λ*Θ*.* Θ nil V E*x.* E(S*x*Θ) ≤ *λ*Θ*. σ.*

This amounts to saying that Φ(*λ*Θ*. σ*) ≤ (*λ*Θ*. σ*), *i.e.* that *λ*Θ*.σ* is a pre- fixed point of Φ; then by Proposition [4.4](#_bookmark26), the *least* fixed point E∞ satisfies E∞ ≤ *λ*Θ*. σ*.

Lemma 4.7 E∞Θ*L* ≤ E∞Θ.

Proof First, E0Θ*L* = Θ nil Λ [*L*]⊥ ≤ Θ nil = E0Θ ≤ E∞Θ, so

E0 ≤ *λ*Θ*L.* E∞Θ. Now suppose that E ≤ *λ*Θ*L.* E∞Θ. Then

$EΘ*L* = Θ nil Λ [*L*]⊥ V E*x.* E(S*x*Θ)(S*xN, P* ) Λ *P* {*x*}

≤ E∞Θ V E*x.* E∞(S*x*Θ) = ΦE∞Θ = E∞Θ*.*

Again we have shown that *λ*Θ*L.* E∞Θ is a pre-fixed point of $, so it is greater than the least fixed point, E∞.

Lemma 4.8 E∞ = E∞ · E∞.

Proof First, E0Θ = Θ nil = E0Θ nil = E0(E0Θ) ≤ E∞(E∞Θ).

Now suppose that E ≤ E∞ · S∞. Then

ΦEΘ = Θ nil V E*x.* E(S*x*Θ)

≤ E∞(S∞Θ) V E*x.* E∞ S∞(S*x*Θ) hypothesis

≤ E∞(S∞Θ) V E*x.* E∞ S*x*($S∞Θ) Definition [4.5](#_bookmark27)

= ΦE∞($S∞Θ) = E∞(S∞Θ)*,*

*i.e.* E∞ · S∞ is a pre-fixed point of Φ, so E∞ ≤ E∞ · S∞, but they are equal since S∞ ≤ id.

Proposition 4.9 K*X* ⊂ Ω is overt (not open), with existential quantifier and

* -modal operator

EK*X* ≡ E · *I* E Σ!K*X* and E = EK*X* · Σ*i.*

Proof We require *θ* ≤ *λl. σ* iff E*l. θl* ≤ *σ*. Forwards, Θ ≤ *λL. σ* ▶ E∞Θ ≤ *σ* (Lemma [4.6](#_bookmark28)). Conversely, S∞Θ*L* ≤ E∞Θ = E∞(S∞Θ) (Lemmas [4.7](#_bookmark29) and [4.8](#_bookmark30)). From this we also have EK*X* · Σ*i* = E∞ · *I* · Σ*i* = E∞ · S∞ = E∞.

Besides equational and fixed point induction (Axiom [2.6](#_bookmark10) and Proposi- tion [4.4](#_bookmark26)), we shall need two other principles in the rest of the paper. The idea of the first one is that ΘT*L* means E*l. L* ~ *l*, and in fact ΘT = S∞T.

Lemma 4.10 Let ▶ ΘT : ΣΩ be the least solution of

ΘT(*N, P* ) = *N* ⊥V E*x.* ΘT(£*xN, P* ) Λ *P* {*x*}*.*

Then *L* : K*X* ▶ ΘT*L* = T, *i.e.* ΘT(*N, P* ) = T for all admissible (*N, P* ).

Proposition 4.11 We have the induction scheme

Γ*, L* : Ω*,* Θ*L* = T ▶ Θ(*x* :: *L*) = T

=============================

Γ ▶ Θ : ΣΩ Γ ▶ Θ nil = T

Γ*, L* : K*X* ▶ Θ*L* = T

Γ ▶ Θ ≤ S*x*Θ

Proof

Θ(*N, P* ) ≥ $S∞Θ(*N, P* )

= S0Θ(*N, P* ) V E*x.* S∞(S*x*Θ)(£*xN, P* ) Λ *P* {*x*}

≥ *N* ⊥ V E*x.* S∞Θ(£*xN, P* ) Λ *P* {*x*}

but ΘT is the least (pre)fixed point of this, so ΘT ≤ Θ. Hence if (*N, P* ) is admissible then Θ(*N, P* ) ≥ ΘT(*N, P* ) = T.

Although the form of this result is very familiar, its usefulness is rather limited, as it only tells us about *open* subspaces. Ultimately, Corollary [10.12](#_bookmark94) will provide an induction scheme of the form that *any* subspace that includes nil and is closed under :: is the whole space. The principle that we shall invoke repeatedly in Section [6](#_bookmark47) is the following.

Proposition 4.12 Let Γ ▶ Θ*,* Φ : ΣΩ (*i.e.* particular Θ and Φ are given). Then

Γ*, L* : Ω ▶ S0Θ*L* ≤ Φ*L*

Γ*,* S ≤ id nucleus*, L* : Ω*,* SΘ*L* ≤ Φ*L* ▶ $SΘ*L* ≤ Φ*L*

Γ*, L* : K*X* ▶ Θ*L* ≤ Φ*L*

*i.e.* Θ*L* ≤ Φ*L* for all admissible *L*.

Proof By equational induction (Axiom [2.6](#_bookmark10)), Γ*, n* : N ▶ $*n*S0Θ ≤ Φ, so Γ ▶ S∞Θ = E*n.* $*n*S0Θ ≤ Φ. Hence if *L* is admissible then Θ*L* = S∞Θ*L* ≤ Φ*L*.

# Stages in the construction

Definition 5.1 Recall from [[B](#_bookmark110)] that Γ ▶ S : ΣΩ → ΣΩ is a *nucleus* on Ω if Γ*,* H : Σ3Ω ▶ S *λL*:Ω*.* H(*λ*Θ:ΣΩ*.* (SΘ*L*)) *L* = S *λL*:Ω*.* H(*λ*Θ:ΣΩ*.* Θ*L*) *L.*

Beware that we have shamelessly appropriated this word from locale theory, in which a nucleus *j* satisfies id ≤ *j* ≤ *j*2. Nuclei in ASD need not in general be order-related to id, but those in this paper will satisfy S ≤ id.

A nucleus Γ ▶ S : ΣΩ → ΣΩ on Ω *admits* a term Γ ▶ *L* : Ω if

Γ*,* Θ : ΣΩ ▶ SΘ*L* = Θ*L* or Γ ▶ *λ*Θ*.* SΘ*L* = *λ*Θ*.* Θ*L.*

If the context Γ is empty, [[B](#_bookmark110)] allows us to introduce a subtype *i* : {Ω | S} > Ω with Σ-splitting *I* for which S = *I* · Σ*i*. Then *L* belongs to the subtype iff S admits it. In this paper we shall need to define nuclei in non-trivial contexts,

but we will not introduce the corresponding dependent types.

Definition 5.2 Following Remark [4.3](#_bookmark25), we construct the subspace K*X* ⊂ Ω using the nucleus S∞ that is defined as the least fixed point of the operator

$ : (ΣΩ → ΣΩ) → (ΣΩ → ΣΩ), where

$SΘ(*N, P* ) = S0Θ(*N, P* ) V E*x.* S(S*x*Θ)(£*xN, P* ) Λ *P* {*x*}

and S0 ≡ $⊥ ≡ *λ*Θ*L.* [*L*]⊥Λ Θ nil ≡ *λ*Θ*NP. N* ⊥Λ Θ(T*,* ⊥)*.*

The objects *X*(*n*) ≡ {Ω | $*n*S0} that are obtained as the successive unwindings of this equation intuitively represent the collections of subsets of *X* with at

most *n* elements.

Lemma 5.3 JT Λ *F* ⊥ ≤ J *F* .

Proof Apply Axiom [2.4](#_bookmark8) to *F* ⊥ = T ▶ T = *F* ▶ JT = J *F* .

This result shows how the singleton { nil } is embedded as a subspace of Ω.

Lemma 5.4 The singleton *i* : { nil }) ) Ω is a Σ-split subspace, where

*σ* : Σ{ nil } ▶ *Iσ* ≡ *λNP. σ* Λ *N* ⊥

Θ : ΣΩ ▶ S0Θ ≡ *I*(Σ*i*Θ) = *λNP.* Θ nil Λ *N* ⊥*.*

Moreover S0 ≤ id, so *I* E Σ*i*, and if S is any nucleus on Ω that admits nil then

S0 ≤ S.

Lemma 5.5 Θ(*N, x* :: *P* ) Λ *P* {*x*} = Θ(*N, P* ) Λ *P* {*x*}

(*x* :: £*xN* ) = (*x* :: *N* ) ≤ *N* ≤ £*xN* = £*x*(*x* :: *N* )

*P* ≤ (*x* :: *P* ) (*x* :: *P* ){*x*} = T *P* {*x*}Λ *φx* ≤ *Pφ*

Proof Apply Axiom [2.4](#_bookmark8) to

*φx* = T ▶ *Nφ* = £*xNφ,* {*x*} ≤ *φ* and *P* {*x*} = T ▶ *x* ::: *P* = *P.*

Lemma 5.6 S(S*x*Θ) ≤ S*x*($SΘ).

Lemma 5.7 If S ≤ id then $S ≤ id.

Lemma 5.8 If S is a nucleus with S ≤ id then so is $S.

Proof Expanding the definition of the *outer* $S, in which H : Σ3Ω,

$S *λL.* H(*λ*Θ*.* ($SΘ*L*)) (*N, P* )

= S0 *λL.* H(*λ*Θ*.* ($SΘ*L*))

V E*x.* S S*x*(*λL.* $SH(*λ*Θ*.* Θ*L*)) (£*xN, P* ) Λ *P* {*x*}*,*

and we have to show that we may delete the *inner* ones. First note that Θ nil ≤ $SΘ nil ≤ Θ nil , by Definition [5.2](#_bookmark36) and Lemma [5.7](#_bookmark40). Then

S0(*λL*'*.* H(*λ*Θ*.* $SΘ*L*'))

= *λL.* *λL*'*.* H(*λ*Θ*.* $SΘ*L*') nil Λ [*L*]⊥

= *λL.* H(*λ*Θ*.* $SΘ nil Λ [*L*]⊥)

= *λL.* H(*λ*Θ*.* Θ nil ) Λ [*L*]⊥ above

= *λL.* *λL*'*.* H(*λ*Θ*.* Θ*L*') nil Λ [*L*]⊥

= S0 *λL*'*.* H(*λ*Θ*.* Θ*L*')

S S*x*(*λL.* H(*λ*Θ*.* Θ*L*))

≥ S S*x*(*λL.* H(*λ*Θ*.* $SΘ*L*)) $S ≤ id

= S *λL.* H(*λ*Θ*.* $SΘ(*x* :: *L*))) def S*x*

≥ S *λL.* H(*λ*Θ*.* S(S*x*Θ)*L*) Lemma [5.6](#_bookmark39)

= S *λL.* (H · ΣS*x*)(*λ*Θ*.* SΘ*L*) def Σ(−)

= S *λL.* (H · ΣS*x*)(*λ*Θ*.* Θ*L*) S nucleus wrt H · ΣS*x*

= S *λL.* H(*λ*Θ*.* (S*x*Θ)*L*) def Σ(−)

= S *λL.* H(*λ*Θ*.* Θ(*x* :: *L*)) def S*x*

= S S*x*(*λL.* H(*λ*Θ*.* Θ*L*)) def S*x*

As we have said, the ASD calculus does not currently allow us to define *X*(*n*) as a *type* dependent on *n*, but we can at least introduce $*n*S0 as a *term* dependent on *n*.

Proposition 5.9 *n* : N ▶ $*n*S0 : ΣΩ → ΣΩ are nuclei with *n* : N ▶ $*n*S0 ≤

$*n*+1S0 ≤ id.

Proof The term *n* : N ▶ $*n*S0 is formed by recursion. The base case of the induction is that ▶ S0 ≤ id is a nucleus, which is Lemma [5.4](#_bookmark37). The induction step,

S : (ΣΩ)(ΣΩ)*,* S ≤ id*,* S nucleus ▶ S ≤ $S ≤ id*,* $S nucleus*,*

was proved in Lemmas [5.7](#_bookmark40) and [5.8](#_bookmark41); it has equations as hypotheses and con- clusions. From these things we may deduce the result by equational induction on N (Axiom [2.6](#_bookmark10)).

Proposition 5.10 ▶ S∞ ≡ E*n.* $*n*S0 is a nucleus with S∞ ≤ id.

Proof By the previous result, $*n*S0 = $*n*+1⊥ is an ascending chain of nuclei with $*n*S0 ≤ id. Writing

S : (ΣΩ)(ΣΩ) ▶ *F* S *, G*S : Σ

for the two sides of Definition [5.1](#_bookmark35) for a nucleus, Axiom [1.9](#_bookmark3) gives

▶ *F* S∞ = *F* (E*n.* $*n*S0) = E*n. F* ($*n*S0)

= E*n. G*($*n*S0) = *G*(E*n.* $*n*S0) = *G*S∞*,*

so S∞ is a nucleus. Lemma [5.7](#_bookmark40) showed that $id ≤ id, so id is a pre-fixed point of $, whence S∞ ≤ id by Proposition [4.4](#_bookmark26).

This justifies Definition [4.5](#_bookmark27) and so the other results of the last section, apart from showing that all definable finite subsets give rise to admissible pairs.

Lemma 5.11 $S admits nil , since Θ nil ≤ $SΘ nil ≤ Θ nil .

Lemma 5.12 If S admits *L* then $S admits *x* :: *L*.

Proof

Θ(*x* :: *L*) = (S*x*Θ)*L* = S(S*x*Θ)*L*

≤ S*x*($SΘ)*L* = $SΘ(*x* :: *L*)

≤ (S*x*Θ)*L.*

So Θ : ΣΩ ▶ Θ(*x* :: *L*) = SΘ(*x* :: *L*) as required.

Lemma 5.13 If ▶ S ≤ id is a nucleus then the pair

1 ni)l

{Ω | $S} (:: *X* × {Ω | S}

is jointly Σ-split epi.

Proof We have just shown that the two maps are well defined. In the diagram

{Ω | $S} (( nil *,* :: 1 + *X* × {Ω | S}

) Σ nil *,* :: )

Σ{Ω|$E} (( Σ × Σ*X*×{Ω|E}

v v

ˆ

ˆ

ˆ ˆ

$S id × S *X*

v Θ '→ (Θ nil *, λxL.* S(S*x*Θ)*L*) ) v

ΣΩ ( Σ × Σ*X*×Ω

*λL. σ* Λ [*L*]⊥V E*x.* Φ*x*(£*x*[*L*]*,* ⟨*L*⟩) Λ ⟨*L*⟩{*x*} ←' (*σ,* Φ)

we have to show that the inverse image map is given by the formula shown, and is split mono.

Let Θ : Σ{Ω|$E}, *x* : *X* and *L* : {Ω | S}, *i.e.* S admits *L*. This means that Θ(*x* :: *L*) = S*x*Θ*L* = S(S*x*Θ)*L* = Φ*xL,*

which justifies the inverse image map. Now the composite takes

Θ '→ (*σ,* Φ) '→ *λL. σ* Λ [*L*]⊥V E*x.* Φ*x*(£*x*[*L*]*,* ⟨*L*⟩) Λ ⟨*L*⟩{*x*}

= *λL.* Θ nil Λ [*L*]⊥V E*x.* S(S*x*Θ)(£*x*[*L*]*,* ⟨*L*⟩) Λ ⟨*L*⟩{*x*}*,*

which is $S Θ, but the hypothesis on Θ says that Θ = $SΘ.

Hence 1 + *X* → *X*(1), 1 + *X* + *X*2 → *X*(2), 1 + *X* + *X*2 + *X*3 → *X*(3), ...

Proposition 5.14 The pair 1 ni)l

K*X* (:: *X* ×K*X* is jointly Σ-split epi.

# Admissible implies modal

Now we use the induction scheme in Proposition [4.12](#_bookmark34) to prove that all admissi- ble pairs Γ ▶ (*N, P* ) : K*X* satisfy the modal laws, starting with the properties of the “possibility” operator *P* .

Lemma 6.1 Σ*X a* ΣΣ*X* by *π* '→ *λφ.* E*x. φx* Λ *πx* and *P* '→ *λx. P* {*x*}.

Indeed, if (*N, P* ) is admissible then *P* is recovered from *π*:

Lemma 6.2

*L* : K*X, φ* : Σ*X* ▶ ⟨*L*⟩*φ* = E*x.* ⟨*L*⟩{*x*}Λ*φx* ≡ E*x. πx*Λ*φx* ≡ E*x. x* ∈ *L*Λ*φx.*

Proof (E*x. P* {*x*}Λ *φx*) ≤ *Pφ* by Lemma [5.5](#_bookmark38), so we have to prove ≥.

Consider Θ ≡ *λNP. Pφ* and Φ ≡ *λNP.* E*y. P* {*y*} Λ *φy* in the context Γ ≡ [*φ* : Σ*X* ]. Then ▶ S0Θ ≤ Φ and S *,* SΘ ≤ Φ ▶ $SΘ ≤ Φ. Hence *L* : K*X, φ* : Σ*X* ▶ ⟨*L*⟩*φ* ≡ Θ*L* ≤ Φ*L* ≡ E*x.* ⟨*L*⟩{*x*} Λ *φx* by Proposition [4.12](#_bookmark34).

Corollary 6.3 ⟨*L*⟩⊥ = ⊥, ⟨*L*⟩(*φ* V *ψ*) = ⟨*L*⟩*φ* V⟨*L*⟩*ψ* ⟨*L*⟩(*φ* Λ *π*) = ⟨*L*⟩*φ*.

Next we consider the “necessity” operator *N* .

Lemma 6.4 (*N, P* ) : K*X* ▶ *Nπ* ≡ *N* (*λx. P* {*x*}) = T, so (*N, P* ) ~ (*N, P* ).

Proof Consider Θ(*N, P* ) = *N* (*λx. P* {*x*}). Then Θ nil = T and (S*x*Θ)(*N, P* ) = (*x* :: *N* ) *λy.* (*x* :: *P* ){*y*}

= *N* (*λy. P* {*y*}V *y* = *x*) Λ (*P* {*x*}V *x* = *x*)

≥ *N* (*λy. P* {*y*}) = Θ(*N, P* )*,*

so by Proposition [4.11](#_bookmark32), *H*(*N, P* ) = T for all admissible (*N, P* ).

Lemma 6.5 If *L* : K*X* then [*L*] preserves finite meets.

Proof We can show that *N* T = T using Θ(*N, P* ) ≡ *N* T in Proposition [4.11](#_bookmark32). For *Nφ* Λ *Nψ* ≤ *N* (*φ* Λ *ψ*) consider Θ(*N, P* ) ≡ *Nφ* Λ *Nψ* and Φ(*N, P* ) ≡ *N* (*φ* Λ *ψ*) in Proposition [4.12](#_bookmark34).

Corollary 6.6 [*L*]T = T, [*L*](*φ* Λ *ψ*) = [*L*]*φ* Λ [*L*]*ψ* and [*L*](*φ* Λ *π*) = [*L*]*φ*.

Proof The last uses Lemma [6.4](#_bookmark50).

Lemma 6.7 *L* : K*X, x* : *X, φ* : Σ*X* ▶ *x* ∈ *L* Λ [*L*]*φ* ≤ *φx*, where (*x* ∈ *L*) ≡ ⟨*L*⟩{*x*}.

Proof With *L* = (*N, P* ), this says that *Nφ* Λ *P* {*x*} ≤ *φx*.

Consider Θ(*N, P* ) = *Nφ* Λ *P* {*x*} and Φ(*N, P* ) = *φx* in Proposition [4.12](#_bookmark34).

Corollary 6.8 [*L*](*φ* V *ψ*) ≤ [*L*]*φ* V ⟨*L*⟩*ψ* and ⟨*L*⟩(*φ* Λ *ψ*) ≥ ⟨*L*⟩*φ* Λ [*L*]*ψ*.

Theorem 6.9 K*X* is overt discrete and has no proper open subalgebra for

nil and ::.

Proof We have just shown that all *L* : K*X* are modal. For modal *L*1*, L*2, Proposition [3.10](#_bookmark19) said that *L*1 = *L*2 : Ω iff (*L*1 ~ *L*2) = T, which is an open equivalence relation. Proposition [4.9](#_bookmark31) said that K*X* is overt. Lemmas [5.11](#_bookmark43)–[5.12](#_bookmark44)

provided the algebra structure and Corollary [4.11](#_bookmark32) said that this is minimal.

Unfortunately + is missing: we are not yet in a position to show that *L*1 + *L*2 is *admissible* when *L*1 and *L*2 are. This is because if *L*1 : *X*(*n*) and *L*2 : *X*(*m*) we would expect *L*1 + *L*2 : *X*(*n*+*m*), whereas our method of induction only takes us from *n* to *n* + 1. We shall show that + is well defined in Lemma [10.9](#_bookmark90), using the recursion scheme for nil and ::, but that is still a long way ahead.

# K as a functor

We must show how K acts on *f* : *X* → *Y* between overt discrete objects, but we shall also prove that it preserves monos and inverse images, and that it takes coproducts (*X* = *Y* + *Z*) to products (K*Y* × K*Z*). This means that any Kuratowski-finite subset *L* : K*X* may be *partitioned* between *Y* and *Z*. (We

already know from Corollary [3.13](#_bookmark23) that K0 = 1.)

Remark 7.1 Recall that *f* : *X* → *Y* acts contravariantly on predicates, turn- ing *ψ* : Σ*Y* into Σ*f ψ* ≡ *λx. ψ*(*f x*) : Σ*X* , but covariantly on modal operators, so *P* : ΣΣ*X* becomes

Σ2*fP* ≡ ΣΣ*f P* ≡ *P* · Σ*f* ≡ *λψ. P* (Σ*f ψ*) ≡ *λψ. P* *λx. ψ*(*f x*) *,*

*i.e.* the composite with the *inverse* image Σ*f* along *f* . In the case of an inclusion *i* : *Y ‹*→ *X*, we may similarly compose with the *direct* image E*i* E Σ*i*, *i.e.*

(*P* · E*i*)*ψ* ≡ *P* (*λx.* E*y. x* = *iy* Λ *ψy*)*.*

In the case of the coproduct, *Y* ⊂ *X* = *Y* + *Z* is also closed, with Σ*i* E 6*i*. For

*ψ* : Σ*Y* and *y* : *Y* , 6*iψ*(*iy*) = *ψy*, whilst for *z* : *Z*, 6*iψ*(*jz*) = T, so

(*N* · 6*i*)*ψ* = *N* *λx.* (E*y. x* = *iy* Λ *ψy*) V (E*z. x* = *jz*) *.*

Locale theorists will recognise this construction as a special case of the rela- tionship with open proper maps that we shall discuss in Section [11](#_bookmark95).

Lemma 7.2 If Γ ▶ (*N, P* ) : Ω*X* is modal then so are Γ ▶ K*f* (*N, P* ) ≡

(*N* · Σ*f , P* · Σ*f* ) and K*i*−1(*N, P* ) ≡ (*N* · 6*i, P* · E*i*) : Ω*Y* .

But we have to show that they are *admissible* to K*Y* whenever *L* : K*X*. Writing Y(*N, P* ) for either of them, we shall do the two cases in parallel, the analogy being that the second is for a *partial* map *f* : *X ~ Y* with decidable support (the “inverse” of *i* : *Y ‹*→ *X*). In Lemma [7.9](#_bookmark61) we shall have to expand an E*x* in the definition of $S*X*; the second case has two sub-cases, in which *x* = *iy* (so *y* = *f x*) and *x* = *jz* (so *fx* is undefined). The following lemmas

therefore have *three* cases. We write S(*n*) = $*n*S

*X*

*X*

0

and S(∞) = E*n.* $*n*S0

for the

nuclei on Ω*X* and similarly for *Y* .

In all three cases, Y preserves the empty subset and unions:

Lemma 7.3 Y nil *X* = nil *Y* , so [*L*]⊥ ≤ [Y*L*]⊥ and S (0)(Θ · Y) ≤ (S (0)Θ) · Y.

*X Y*

Proof Σ*f* T = 6*i*T = T and Σ*f* ⊥ = E*i*⊥ = ⊥.

[Y*L*]⊥ = [*L*](*H*⊥) ≥ [*L*]⊥ where *H* = Σ*f* or 6*i*.

S (0)(Θ · Y)*L* = Θ(Y nil ) Λ [*L*]⊥ ≥ Θ nil Λ [Y*L*]⊥ = (S (0)Θ)(Y*L*).

*X Y*

Lemma 7.4 *L*1*, L*2 : Ω*X* ▶ Y(*L*1 + *L*2) = Y*L*1 + Y*L*2,

so + on Ω is natural in *f* : *X* → *Y* .

Proof (*N*1 Λ *N*2) · *H* = *N*1 · *H* Λ *N*2 · *H* and (*P*1 V *P*2) · *H* = *P*1 · *H* V *P*2 · *H*,

where *H* = Σ*f* , 6*i* or E*i*.

Y either applies the function *f* to individual elements *x*, or “filters” the defined values *y* = *fx* = *f* (*iy*) from the undefined ones with *x* = *jz*.

Lemma 7.5 Y{|*x*|} = {|*f x*}| , Y{|*iy*}|

= {|*y*}|

and Y{|*jz*}|

= nil .

Proof We do , the proof for ♦ being similar.

[{|*x*|}](Σ*f ψ*) = *ηx*(Σ*f ψ*) = (Σ*f ψ*)*x* = *ψ*(*f x*) = *η*(*f x*)*ψ* = [{|*f x*|}]*ψ*

[{|*iy*|}](6*iψ*) = *η*(*iy*)(6*iψ*) = (6*iψ*)(*iy*) = *ψy* = *ηyψ* = [{|*y*}| ]*ψ*

[{|*jz*}| ](6*iψ*) = *η*(*jz*)(6*iψ*) = (6*iψ*)(*jz*) = T = [ nil ]*ψ*.

Corollary 7.6 S*x*(Θ · Y) = S(*f x*)Θ · Y, S(*iy*)(Θ · Y) = S*y*Θ · Y and

S(*jz*)(Θ · Y) = Θ · Y.

Proof Θ · Y({|*x*|} + *L*) = Θ(Y{|*x*|} + Y*L*), which is respectively

Θ({|*f x*}| + Y*L*), Θ({|*y*}| + Y*L*) and Θ( nil + Y*L*).

Lemma 7.7 *P* {*x*} ≤ (*P* · Σ*f* ){*f x*}, *P* {*iy*} = (*P* · E*i*){*y*} and

*P* {*jz*} = (*P* · E*j*){*z*}.

Proof {*x*}*x*' = (*x* = *x*') ≤ (*fx* = *f x*') = Σ*f* {*f x*}*x*'

and {*iy*}*x*' = (*iy* = *x*') = E*y*'*.* (*y* = *y*') Λ (*iy*' = *x*') = E*i*{*y*}*x*'.

Lemma 7.8 “*fL* \ *fx* ⊂ *f* (*L* \ *x*)” in the sense that

(£*xN* ) · Σ*f* ≤ £(*f x*)(*N* · Σ*f* ), so Y(£*xN, P* ) ≤ £(*f x*) Y(*N, P* ) ,

(£(*iy*)*N* ) · 6*i* = £*y*(*N* · 6*i*), so Y £(*iy*)*N, P* = £*y* Y(*N, P* ) ,

and (£(*jz*)*N* ) · 6*i* = (*N* · 6*i*), so Y £(*jz*)*N, P* = Y(*N, P* ).

Proof *f*

(£*xN* ) · Σ

*ψ* = *N*

' ' '

*λx .x* = *x* V *ψ*(*fx* )

≤ *N* *λx*'*. fx*' = *fx* V *ψ*(*f x*')

= *N* Σ*f* (*λy. y* = *fx* V *ψy*)

= £(*f x*)(*N* · Σ*f* ) *ψ*

(£(*iy*)*N* ) · 6*i* *ψ* = *N* (*λx. x* = *iy* V 6*iψx*)

= *N* *λx.* E*y*'*.x* = *iy*' Λ (*iy*' = *iy* V 6*iψ*(*iy*'))

V E*z*'*.x* = *jz*' Λ (*jz*' = *iy* V 6*iψ*(*jz*'))

= *N* *λx.* E*y*'*.x* = *iy*' Λ (*y*' = *y* V 6*iψy*') V E*z*'*.x* = *jz*' Λ (⊥V T)

= (*N* · 6*i*)(*λy*'*.y* = *y*' V *ψy*')

= £*y*(*N* · 6*i*)*ψ*

(£(*jz*)*N* ) · 6*i* *ψ* = *N* (*λx. x* = *jz* V 6*iψx*)

= *N* *λx.* E*y*'*.x* = *iy*' Λ (*iy*' = *jz* V 6*iψ*(*iy*'))

V E*z*'*.x* = *jz*' Λ (*jz*' = *jz* V 6*iψ*(*jz*'))

= *N* *λx.* E*y*'*.x* = *iy*' Λ (⊥V *ψy*')

V E*z*'*.x* = *jz*' Λ (*z* = *z*' V T)

= *N* *λx.* E*y*'*.x* = *iy*' Λ *ψy*') V E*z*'*.x* = *jz*'

= (*N* · 6*i*)*ψ*

Lemma 7.9 *λ*Φ*.* S*X*(Φ · Y) ≤ *λ*Φ*.* (S*Y* Φ) · Y ▶ *λ*Θ*.* $S*X*(Θ · Y) ≤

*λ*Θ*.* ($S*Y* Θ) · Y.

Proof We expand $S*X* (Definition [5.2](#_bookmark36)). By Lemma [7.3](#_bookmark56),

$S*X*(Θ · Y)(*N, P* ) = S (0)(Θ · Y)(*N, P* ) V E*x.* (·· ·)

*X*

≤ (S (0)Θ) Y(*N, P* ) V E*x.* (·· ·)*,*

*Y*

where the term (·· ·) has three cases:

In the first, for *f* , use Corollary [7.6](#_bookmark58) and the premise with Φ ≡ S(*f x*)Θ:

S*X* S*x*(Θ · Y) ≤ S*X* (S(*f x*)Θ) · Y ≤ S*Y* (S(*f x*)Θ) · Y*.*

Hence, using Lemmas [7.7](#_bookmark59) and [7.8](#_bookmark60), and putting *y* ≡ *f x*, (·· ·) ≡ S*X* S*x*(Θ · Y) (£*xN, P* ) Λ *P* {*x*}

≤ S*Y* (S(*f x*)Θ) Y(£*xN, P* ) Λ (*P* · Σ*f* ){*f x*}

≤ S*Y* (S*y*Θ) £*y*(Y(*N, P* )) Λ ⟨Y(*N, P* )⟩{*y*}

≤ $S*Y* Θ Y(*N, P* ) def $S*Y*

Similarly, S*X* S(*iy*)(Θ · Y) ≤ S*X* (S*y*Θ) · Y ≤ S*Y* (S*y*Θ) · Y, so (·· ·) ≡ S*X* S(*iy*)(Θ · Y) (£(*iy*)*N, P* ) Λ *P* {*iy*}

≤ S*Y* (S*y*Θ) Y(£(*iy*)*N, P* ) Λ (*P* · E*i*){*y*}

≤ S*Y* (S*y*Θ) £*y*(Y(*N, P* )) Λ ⟨Y(*N, P* )⟩{*y*}

≤ $S*Y* Θ Y(*N, P* )

Finally, S*X* S(*jz*)(Θ · Y) ≤ S*X*(Θ · Y) ≤ (S*Y* Θ) ·Y ≤ ($S*Y* Θ) · Y, so (·· ·) ≡ S*X* S(*jz*)(Θ · Y) (£(*jz*)*N, P* ) Λ *P* {*jz*}

≤ $S*Y* Θ Y(£(*jz*)*N, P* )

≤ $S*Y* Θ Y(*N, P* )

Proposition 7.10 If *L* : K*X* then Y*L* : K*Y* in all three cases.

Proof By Lemma [7.3](#_bookmark56), ▶ *λ*Θ*.* S (0)(Θ · Ω*f* ) ≤ *λ*Θ*.* (S (0)Θ) · Ω*f* , and

*X Y*

*n* : N*, λ*Φ*.* S(*n*)(Φ · Y) ≤ *λ*Φ*.* (S(*n*)Φ) ·Y

*X Y*

*λ*Θ*.* S(*n*+1)(Θ · Y) ≤ *λ*Θ*.* (S(*n*+1)Θ) · Y*,*

*X Y*

by Lemma [7.9](#_bookmark61). So by equational induction (Axiom [2.6](#_bookmark10)),

*n* : N ▶ *λ*Θ*.* S(*n*)(Θ · Ω*f* ) ≤ *λ*Θ*.* (S(*n*)Θ) · Ω*f ,*

*X Y*

whence Θ : ΣΣ*Y* ▶ S(∞)(Θ · Ω*f* ) ≤ (S(∞)Θ) · Ω*f .*

*X Y*

As Γ ▶ *L* : K*X* = {Ω*X* | S(∞)}, we have Γ*,* Φ : ΣΩ*X* ▶ Φ*L* = S(∞)Φ*L*.

*X* *X*

Putting Φ = Θ · Y,

Θ(Y*L*) = Φ*L* = (S(∞)Φ)*L* = S(∞)(Θ · Y)*L* ≤ (S(∞)Θ)(Y*L*) ≤ Θ(Y*L*)*,*

*X X Y*

so these are equal and Γ ▶ Y*L* : {Ω*Y* | S(∞)} = K*Y* .

*Y*

We exploit the first case to define K as a functor.

Theorem 7.11 Let *f* : *X* → *Y* between overt discrete spaces and *L* : K*X*.

1. Then K*fL* ≡ Y*L* : K*Y* ,
2. the inclusion K(−) > Ω(−) is natural in *f* ,
3. K*f* is a homomorphism in the sense that K*f* nil *X* = nil *Y* and

K*f* (*x* :: *L*) = *fx* :: K*f L*;

1. K is a covariant functor, *i.e.* Kid*X* = idK*X* and K(*g* · *f* ) = K*g* · K*f* .
2. In fact, K*f* (*L* + *L*') = K*fL* + K*f L*', but we don’t yet know that this is admissible.

Proof [a] Proposition [7.10](#_bookmark62), [b] Lemma [7.3](#_bookmark56) and Corollary [7.6](#_bookmark58), [c] Remark [7.1](#_bookmark55) and [d] Lemma [10.9](#_bookmark90).

The other two cases almost show that K(*Y* + *Z*) ~= K*Y* × K*Z*.

Lemma 7.12 If (*N, P* ) : Ω*X* , (*N*1*, P*1) : Ω*Y* and (*N*2*, P*2) : Ω*Z* are modal then the following are inverse:

(*N*1*, P*1)*,* (*N*2*, P*2) ) (*N*1 · Σ*i* Λ *N*2 · Σ*j, P*1 · Σ*i* V *P*2 · Σ*j*)

(*N* · 6*i,P* · E*i*)*,* (*N* · 6*j,P* · E*j*) ( (*N, P* )

If (*N, P* ) is admissible then so are (*N*1*, P*1) and (*N*2*, P*2). Conversely, if

(*N*1*, P*1) = {|*y*}| and (*N*2*, P*2) is admissible then so is (*N, P* ) = *iy* :: K*j*(*N*2*, P*2).

Proof The isomorphism uses the equations

6*i* · Σ*i* Λ 6*j* · Σ*j* = idΣ2 *X* = 6*i* · Σ*i* Λ 6*j* · Σ*j*

Σ*i* · 6*i* = idΣ*Y* = Σ*i* · E*i* Σ*j* · 6*j* = idΣ*Z* = Σ*j* · E*j*

Σ*i* · 6*j* = T Σ*j* · 6*i* = T Σ*i* · E*j* = ⊥ Σ*j* · E*i* = ⊥

Finally, K*i*{|*y*}| + K*jL* = (*iy*) :: K*jL*.

This is enough for us to proceed to the investigation of List*X*. With the aid of that, we shall later be able to prove that + is admissible, making K*X* a

semilattice — indeed, the free one on *X*. This will also remove the restriction in the previous result.

We conclude this section by considering a *non-complemented* inclusion *i* : *U* ⊂ *X*. In this case we no longer have 6*i*, but we can instead apply E*i* to both *N* and *P* .

Lemma 7.13 Let *i* : *U ‹*→ *X* be a mono between overt discrete spaces, classified by *φ* : *X* → Σ. Then K*i* : K*U* → K*X* is a split mono, classified by

*φ* : K*X* → Σ.

*U*  ) 1 K*U*  ) 1

∩

v

∩

*i* T

v *φ* ) v

K*i* T

*φ* ) v

*X* Σ

Proof K*i* is split mono since

K*i*

K*X* Σ

K∃b

(*N, P* ) ) (*N* · Σ*i,P* · Σ*i*)

)*i* (*N* · Σ*i* ·E *, P* · Σ*i* ·E ) = (*N, P* )*.*

*i* *i*

The square on the right commutes, since

*L* : K*U* ▶ *φ*(K*iL*) = [K*iL*]*φ* = [*L*](Σ*iφ*) = [*L*]T = T*.*

If Γ ▶ *L* : K*X* with *φL* ≡ [*L*]*φ* = T then [*L*](*φ* Λ *ψ*) = [*L*]*φ* Λ [*L*]*ψ* = [*L*]*ψ* and ⟨*L*⟩*ψ* = [*L*]*φ* Λ ⟨*L*⟩*ψ* ≤ ⟨*L*⟩(*φ* Λ *ψ*) ≤ ⟨*L*⟩(*ψ*). Hence

K*i*(KE^*iL*) = ([*L*] · E*i* · Σ*i,* ⟨*L*⟩· E*i* · Σ*i*)

= *λψ.* [*L*](*φ* Λ *ψ*)*, λψ.* ⟨*L*⟩(*φ* Λ *ψ*)

= (*λψ.* [*L*]*ψ, λψ.* ⟨*L*⟩*ψ*) = *L,*

so *L* belongs to the retract, which is K*U* .

Theorem 7.14 K preserves monos and their inverse images.

*U*  ) *V*  ) 1 K*U*  ) K*V*  ) 1

∩

v

∩

v

∩ ∩

*i j*

*f* ) v

T

*φ* ) v

K*i* K*j*

K*f* ) v

T

*φ* ) v

*X Y* Σ K*X* K*Y* Σ

Proof Since *V* and *U* are classified by *φ* and *φ* · *f* , we have just shown that

K*V* and K*U* are classified by *φ* and (*φ* · *f* ), so we just need to check that

(*φ* · *f* ) = *λL.* [*L*](*φ* · *f* ) = *λL.* ([*L*] · Σ*f* )*φ* = *φ* · K*f.*

# Lists, heads and tails

We shall derive the recursive properties of K*X* from those of List*X*. The idea of the representation is that a list or sequence of length *n* is a partial function N *~ X* with support {0*,* 1*,...,n* − 1}, encoded as a finite set of pairs.

Definition 8.1 List*X* ⊂ K(N × *X*) is the open subspace classified by

*λL.* [*L*] *λnx.* [*L*](*λmy. n* /= *m* V *x* = *y*)

Λ [*L*] *λnx. n* = 0 V ⟨*L*⟩(*λmy. n* = *m* + 1) *.*

In more suggestive notation (which will be justified in Lemma [9.10](#_bookmark81)) List*X* consists of the finite sets *L* of pairs such that 6(*n, x*)*,* (*m, y*) ∈ *L. n* /= *m*V*x* = *y* and 6(*n, x*) ∈ *L. n* = 0 V E(*m, y*) ∈ *L. n* = *m* + 1. The first condition says that *L*, considered as a binary relation, is functional, and the second that this

function is defined on an initial segment.

Proposition 8.2 List*X* is overt discrete, being an open subspace of an overt discrete space.

Definition 8.3 For *z* : *X*, *L* : List*X*, let (*z* ::: *L*) ≡ K*i*{|*z*|} + K*jL* and

tail *L* = K*j*−1*L*, where

*X*) *i* = (0*,* id) ) N × *X* ( *j* = succ ×id (N × *X.*

Expanding the definitions from the previous section,

[*z* ::: *L*]*φ* ≡ *φ*(0*, z*) Λ [*L*] *λmy. φ*(*m* + 1*, y*)

⟨*z* ::: *L*⟩*φ* ≡ *φ*(0*, z*) V ⟨*L*⟩ *λmy. φ*(*m* + 1*, y*)

[tail *L*]*φ* ≡ [*L*] *λnx. n* = 0 V E*m. n* = *m* +1 Λ *φ*(*m, x*)

⟨tail *L*⟩*φ* ≡ ⟨*L*⟩ *λnx.* E*m. n* = *m* +1 Λ *φ*(*m, x*) *.*

Lemma 8.4 nil ∈ List*X*, whilst if *z* : *X* and *L* : List*X* then (*z* ::: *L*) : List*X*.

Proof As [ nil ]*φ* = T, the two conditions in Definition [8.1](#_bookmark68) are easily satisfied.

Lemma [7.12](#_bookmark65) is enough to show that *z* ::: *L* is in K(N × *X*). To show that it is in List*X*, we expand first the outer [*z* ::: *L*] and then the inner [*z* ::: *L*] or

⟨*z* ::: *L*⟩, using Corollaries [6.3](#_bookmark49) and [6.6](#_bookmark51) and some basic arithmetic.

[*z* ::: *L*] *λnx.* [*z* ::: *L*](*λmy. n* /= *m* V *x* = *y*)

= [*z* ::: *L*](*λmy.* 0 /= *m* V *z* = *y*)

Λ [*L*] *λnx.* [*z* ::: *L*](*λmy. n* +1 /= *m* V *x* = *y*)

= (0 /= 0 V *z* = *z*) Λ [*L*](*λmy.* 0 /= *m* +1 V *z* = *y*)

Λ [*L*] *λnx. n* +1 /= 0 V *x* = *z*)

Λ [*L*] *λnx.* [*L*](*λmy. n* +1 /= *m* +1 V *x* = *y*)

= TΛ TΛ T Λ [*L*] *λnx.* [*L*](*λmy. n* +1 /= *m* +1 V *x* = *y*)

= [*L*] *λnx.* [*L*](*λmy. n* /= *m* V *x* = *y*) = T

[*z* ::: *L*] *λnx. n* = 0 V ⟨*z* ::: *L*⟩(*λmy. n* = *m* + 1)

= 0 = 0 V ⟨*z* ::: *L*⟩(*λmy.* 0 = *m* + 1)

Λ [*L*] *λnx. n* +1 = 0 V ⟨*z* ::: *L*⟩(*λmy. n* +1 = *m* + 1)

= T Λ [*L*] *λnx.* ⊥V *n* +1 = 0 +1 V ⟨*L*⟩(*λmy. n* +1 = *m* +1+ 1)

= [*L*] *λnx. n* = 0 V ⟨*L*⟩(*λmy. n* = *m* + 1) = T

From this we can show that List is a functor to the category of internal (imposed) monoids.

Theorem 8.5 Let *f* : *X* → *Y* between overt discrete spaces and *L* : List*X*.

Then

List(*f* )(*L*) ≡ K(N × *f* )(*L*) : List(*Y* )*,*

List(*f* ) is a homomorphism for nil and :::, *i.e.*

List(*f* )( nil *X*) = nil *Y* and List(*f* )(*x* ::: *L*) = *fx* ::: List(*f* )(*L*)*,* and List is a covariant functor, *i.e.*

List(id*X* ) = idList(*X*) and List(*g* · *f* ) = List(*g*) · List(*f* )*.*

Functional programmers write map*fL* for List*f L*.

Proof We have K(N × *f* )(*L*) : K(N × *Y* ) by Proposition [7.11](#_bookmark63), so we have to show that it satisfies Definition [8.1](#_bookmark68), and also show that List*f* is a homomor-

phism for :::.

[K(N × *f* )(*L*)] *λny.* [K(N × *f* )(*L*)](*λmy*'*.n* /= *m* V *y* = *y*')

= Σ2(N × *f* )[*L*] *λny.* Σ2(N × *f* )[*L*](*λmy*'*.n* /= *m* V *y* = *y*') [7.11](#_bookmark63)

= [*L*] *λnx.* [*L*](*λmx*'*.n* /= *m* V *fx* = *f x*') Remark [7.1](#_bookmark55)

≥ [*L*] *λnx.* [*L*](*λmx*'*.n* /= *m* V *x* = *x*') = T (*x* = *x*') ≤ (*fx* = *f x*')

[K(N × *f* )(*L*)] *λny. n* = 0 V ⟨K(N × *f* )(*L*)⟩(*λmy*'*.n* = *m* + 1)

= [*L*] *λnx. n* = 0 V ⟨*L*⟩(*λmx*'*.n* = *m* + 1) = T Proposition [7.11](#_bookmark63)

K(N × *f* )(*x* ::: *L*)

= K(N × *f* ) (0*, x*) :: K(succ ×*X*)*L* Definition [8.3](#_bookmark69)

= (0*,fx*) :: K(N × *f* ) · K(succ ×*X*)*L* Proposition [7.11](#_bookmark63)

= (0*,fx*) :: K(succ ×*Y* ) · K(N × *f* )*L* def (succ ×*f* )

= K(succ ×*Y* ) *fx* ::: K(N × *f* )*L* Proposition [7.11](#_bookmark63)

The other parts follow directly from Proposition [7.11](#_bookmark63).

Next we show that every list is either empty or has a unique head and tail.

Lemma 8.6 For *L* : List*X* and *n* : N,

E*x.* (*n* + 1*, x*) ∈ *L* ≤ E*y.* (*n, y*) ∈ *L* ≤ E*z.* (0*, z*) ∈ *L* *.*

Proof Write *αn* ≡ ⟨*L*⟩(*λmx. n* = *m*) ≡ E*x.* (*n, x*) ∈ *L* . The second part of Definition [8.1](#_bookmark68) for *L* : List*X* says

[*L*] *λnx. n* = 0 V ⟨*L*⟩(*λmy. n* = *m* + 1) *,*

which, together with the hypothesis *αn*+1 = T, gives

*n* +1 = 0 V ⟨*L*⟩(*λmy. n* +1 = *m* + 1) = T*,*

which is equivalent to *αn* = T. Hence by Axiom [2.4](#_bookmark8) we have *αn*+1 ≤ *αn* and

*L* : List*X, n* : N*, αn* ≤ *α*0 ▶ *αn*+1 ≤ *α*0*,*

which is the induction step for proving *L* : List*X, n* : N ▶ *αn* ≤ *α*0.

Lemma 8.7 If *L* : List*X* then *L* = nil V E!*z.* (0*, z*) ∈ *L*.

Proof From Proposition [3.12](#_bookmark22), either *L* ~ nil or E*nx.* (*n, x*) ∈ *L*. In the latter case the previous result applies, giving E*z.* (0*, z*) ∈ *L*, which is unique by the first part of Definition [8.1](#_bookmark68).

Lemma 8.8 Let *L* : List*X* and *x* : *X*. Then tail *L* : K(N×*X*), tail (*x* ::: *L*) = *L*, and if (0*, z*) ∈ *L* : List*X* then *L* = *z* ::: tail *L*.

Proof These are corollaries of Lemma [7.12](#_bookmark65).

tail (*z* ::: *L*) = K*j*−1(K*i*{|*z*|} + K*jL*)

= (K*j*−1 · K*i*){|*z*|} + (K*j*−1 · K*jL*) = nil + *L z* ::: tail *L* = K*i*{|*z*|} + K*j*(tail *L*)

= K*i*(K*i*−1*L*)+ K*j*(K*j*−1*L*) = *L* {|*z*|} = K*i*−1*L*

Lemma 8.9 If *L* : List*X* then tail *L* : List*X*.

Proof

[tail *L*] *λnx.* [tail *L*](*λmy. n* /= *m* V *x* = *y*)

= [*L*] *λnx. n* = 0 V E*n*'*.n* = *n*' +1 Λ [*L*](*λmy. m* = 0

V E*m*'*.m* = *m*' +1 Λ (*n*' /= *m*' V *x* = *y*))

≥ [*L*] *λnx. n* = 0 V (*n* /= 0 Λ [*L*](*λmy. n* /= *m* V *x* = *y*))

≥ [*L*] *λnx.* [*L*](*λmy. n* /= *m* V *x* = *y*) ≥ T

[tail *L*] *λnx. n* = 0 V ⟨tail *L*⟩(*λmy. n* = *m* + 1)

= [*L*] *λnx. n* = 0 V E*n*'*.n* = *n*' +1 Λ *n*' = 0

V ⟨*L*⟩(*λmy.* E*m*'*.m* = *m*' +1 Λ *n*' = *m*' + 1)

≥ [*L*] *λnx. n* = 0 V *n* /= 0 Λ *n* = 1 V ⟨*L*⟩(*λmy. n* = *m* +1 /= 1)

≥ [*L*] *λnx. n* = 0 V *n* /= 0 Λ ⟨*L*⟩(*λmy. n* = *m* + 1)

≥ [*L*] *λnx.* ⟨*L*⟩(*λmy. n* = *m* + 1) ≥ T

Proposition 8.10 List*X* ~= 1 + *X* × List*X*, *cf.* Proposition [5.14](#_bookmark46) for K*X*. In particular,

1. *L* : List*X* ▶ (*L* =List*X* nil V E*xL*'*.L* =List*X x* ::: *L*') = T, and
2. *x, y* : *X, L, L*' : List*X* ▶

(*x* ::: *L* =List*X y* ::: *L*') = (*x* =*X y*) Λ (*L* =List*X L*').

# Recursion over lists

Lemma 9.1 Define ↓ : N → List(1) by recursion,

↓ 0 = nil and ↓(*n* + 1) = *\** ::: (↓ *n*)*,*

where *\** : 1, so ↓ is a homomorphism. Then (*k, \**) ∈ ↓ *n* = (*k < n*), [↓ *n*] = *λφ.* 6*k < n. φ*(*k, \**) and ⟨↓ *n*⟩ = *λφ.* E*k < n. φ*(*k, \**)*.*

Proof The recursive definition says that [↓ 0]*φ* = T and [↓(*n* + 1)]*φ* = *φ*(0*, \**) Λ [↓ *n*] *λky. φ*(*k* + 1*, y*) *,*

but *λφ.* 6*k < n. φ*(*k, \**) satisfies the same equations (Remark [3.2](#_bookmark13)), so they are equal by the universal property of N. Similarly for ⟨↓ *n*⟩ with ⊥, V and E in place of T, Λ and 6.

Remark 9.2 This means that

↓ 1 = {|(0*, \**)}| *,* ↓ 2 = {|(0*, \**)*,* (1*, \**)|}*,* ↓ 3 = {|(0*, \**)*,* (1*, \**)*,* (2*, \**)}| *,...*

but we shall ignore the *\** from now on, and regard List(1) ⊂ K(N).

Lemma 9.3 There is a unique map |−| : List(1) → N such that (*n <* |*L*|) = (*n* ∈ *L*) ≡ ⟨*L*⟩(*λx. x* = *n*)

and (|*L*| ≤ *n*) = (*n* ∈*/ L*) ≡ [*L*](*λx. x* /= *n*)*.*

Proof We must define |*L*| in terms of the presence or absence of numbers in the list, as it would be begging the question to use recursion on List(1).

The two properties *n* ∈ *L* and *n* ∈*/ L* (as defined by ≡ above) are comple- mentary by the modal laws, in particular Corollary [6.8](#_bookmark52).

T = [*L*](*λx. x* /= *n* V *x* = *n*)

≤ [*L*](*λx. x* /= *n*) V ⟨*L*⟩(*λx. x* = *n*) ≡ (*n* ∈*/ L*) V (*n* ∈ *L*)

⊥ = ⟨*L*⟩(*λx. x* /= *n* Λ *x* = *n*)

≥ [*L*](*λx. x* /= *n*) Λ ⟨*L*⟩(*λx. x* = *n*) ≡ (*n* ∈*/ L*) Λ (*n* ∈ *L*)*.*

Now *φ* ≡ *λn.* (*x < n*) satisfies the premise of Axiom [1.9](#_bookmark3), so

T = [*L*](*λx.* E*n. x < n*) = E*n.* [*L*](*λx. x < n*)

≤ E*n.* [*L*](*λx. x* /= *n*) = E*n. n* ∈*/ L.*

Hence we may use general recursion or sobriety [[A](#_bookmark109), Lemma 9.11] to define

|*L*| ≡ *µn. n* ∈*/ L,* which therefore satisfies (*n <* |*L*|) ≤ (*n* ∈ *L*) and

(|*L*| ≤ *n*) ≥ (*n* ∈*/ L*). These are actually equalities, as Lemma [8.6](#_bookmark70) said that

(*n* +1 ∈ *L*) ≤ (*n* ∈ *L*)*,* and so (*n* ∈*/ L*) ≤ (*n* +1 ∈*/ L*)*,*

and indeed (*m* ≥ *n* ∈*/ L*) ≤ (*m* ∈*/ L*) by induction.

Proposition 9.4 |−| and ↓ make List(1) =~ N.

Proof We show that ↓ and |−| are inverse. Using Lemmas [9.1](#_bookmark74), [9.3](#_bookmark78) and [6.2](#_bookmark48),

⟨↓ |*L*|⟩*φ* = E*x <* |*L*|*. φx* = E*x. x <* |*L*|Λ *φx*

= E*x.* ⟨*L*⟩(*λy. x* = *y*) Λ *φx* = ⟨*L*⟩*φ,*

so ↓ |*L*| = *L* by Proposition [3.10](#_bookmark19). On the other hand,

(|↓ *m*| *> n*) = ⟨↓ *m*⟩(*λx. x* = *n*) = (E*x < m. x* = *n*) = (*n < m*)

by Lemmas [9.3](#_bookmark78) and [9.1](#_bookmark74), so |↓ *m*| = *m*.

Corollary 9.5 For any overt discrete object *X* there is a map |−| : List*X* → N

such that

|*L*| = 0 ⇐⇒ *L* = nil *,* and |*z* ::: *L*| = |*L*| + 1*.*

The number |*L*| is called the *length* of the list.

Proof Define |*L*| ≡ |List(!*X* )*L*|, which is the composite of one homomor- phism with the inverse of another. By Lemma [8.7](#_bookmark71), (*L* = nil ) is decidable. If *L* /= nil then *L* = *x* ::: *L*' for some *x* and *L*', so |*L*| = |*L*'| +1 /= 0.

Now we can start to define recursion over lists, in a way that will be very familiar to functional programmers. The first result depends on Propo- sition [8.10](#_bookmark75) and establishes existence, and the second uses Corollary [9.5](#_bookmark79) and gives uniqueness.

Lemma 9.6 Let *A* = Σ*U* for some object *U* , equipped with an *action* of *X*: Γ ▶ *ζ* : *A* and Γ*, x* : *X, α* : *A* ▶ *σ*(*x, α*) : *A.*

Then the term Γ ▶ *ϵ* : *A*List*X* is a homomorphism in the sense that

*ϵ* nil = *ζ* and *ϵ*(*x* :: *L*) = *σ*(*x, ϵL*)*.*

iff it satisfies the fixed point equation

*ϵ* = *λL.* (*L* = nil Λ *ζ*) V E*x*:*X.* E*L*' :List*X.* (*L* = *x* ::: *L*') Λ *σ*(*x, ϵL*')*.*

Proof If *ϵ* is a homomorphism then by Proposition [8.10(a)](#_bookmark76),

*ϵL* = (*L* = nil Λ *ϵL*) V E*xL*'*.* (*L* = *x* ::: *L*' Λ *ϵL*)

= (*L* = nil Λ *ζ*) V E*xL*'*.L* = *x* ::: *L*' Λ *σ*(*x, ϵL*')

Conversely, if it is a fixed point then by Proposition [8.10(b)](#_bookmark77),

*ϵ* nil = ( nil = nil Λ *ζ*) V E*xL*'*.* ⊥Λ *σ*(*x, ϵL*') = *ζ*

*ϵ*(*y* ::: *L*) = (*y* ::: *L* = nil Λ *ζ*) V E*xL*'*.* (*y* ::: *L* = *x* ::: *L*') Λ *σ*(*x, ϵL*')

= ⊥ V E*xL*'*.* (*x* = *y*) Λ (*L* = *L*') Λ *σ*(*x, ϵL*')

= *σ*(*y, ϵL*)*.*

The simplicity of the last step is one reason why we have done this part of the argument for lists: the corresponding argument for finite subsets would be much more difficult. The other reason is that lists have a well defined length, which is the key to proving uniqueness, and more generally *equational induction for lists*.

Proposition 9.7 Let Γ*, L* : List*X* ▶ *αL, βL* : *A* ≡ Σ*U* . Then, *cf.* Axiom [2.6](#_bookmark10),

Γ ▶ *α* nil = *β* nil Γ*, L* : List*X, x* : *X, αL* = *βL* ▶ *α*(*x* ::: *L*) = *β*(*x* ::: *L*) Γ*, L* : List*X* ▶ *αL* = *βL*

Proof Consider *αn* ≡ *λL.* |*L*| ≤ *n* Λ *αL* and *βn* ≡ *λL.* |*L*| ≤ *n* Λ *βL*,

both of type Σ*A*×List*X*. Then *α* = *β* because (*L* = nil ) ⇐⇒ (|*L*| = 0) by

0

0

Corollary [9.5](#_bookmark79).

Suppose that *αn* = *βn*. Then if *L*' : List*X* with |*L*'| ≤ *n*, we have *αL*' =

*αnL*' = *βnL*' = *βL*' and so *α*(*x* ::: *L*') = *β*(*x* ::: *L*') by hypothesis. Hence,

using Proposition [8.10(a)](#_bookmark76) and Corollary [9.5](#_bookmark79),

*αn*+1 = *λL.* *L* = nil V E*x.* E*L*'*.* (*L* = *x* ::: *L*') Λ (|*L*| ≤ *n* + 1) Λ *αL*

= *λL.* (*L* = nil Λ *α* nil )

V E*x.* E*L*'*.* (*L* = *x* ::: *L*') Λ (|*L*'| ≤ *n*) Λ *α*(*x* ::: *L*')

= (the same for *β*) = *βn*+1*.*

Thus *αn* = *βn* by equational induction for N (Axiom [2.6](#_bookmark10)), and

*αL* = (|*L*| ≤ |*L*|) Λ *αL* = *α*|*L*|*L* = *β*|*L*|*L* = *βL.*

Theorem 9.8 List*X* is the free (imposed, :::) monoid on *X*. It also obeys equational induction at all types.

Proof Let *M* be any type with an action Γ ▶ *z* : *M, r* : *X* × *M* → *M* . We have already shown that some homomorphism List*X* → *M* exists and is unique in the case where *M* = Σ*U* , so we consider *A* ≡ ΣΣ*M* , *ζ* = *λψ. ψx* : *A* and *σ* : *X* × *A* → *A* by *σ*(*x, F* ) = *λψ. F* *λm.* (*ψ* · *r*)(*x, m*) .

Then there is a unique map *ϵ* : List*X* → *A* such that

*ϵ* nil = *ζ* = *λψ. ψz*

*ϵ*(*x* ::: *L*) = *σ*(*x, ϵL*) = *λψ. ϵL* *λm.* (*ψ* · *r*)(*x, m*) *.*

Now we use Proposition [9.7](#_bookmark80) to show that *ϵL* is prime [[A](#_bookmark109), §4]. Clearly *ϵ* nil is prime. Suppose that *ϵL* is prime, so (with J : Σ3*M* )

*λ*J *.* J(*ϵL*) = *λ*J *. ϵL* *λm.* J(*λψ. φm*) *.*

We shall show that *ϵ*(*x* ::: *L*) is also prime (with respect to G : Σ3*M* ), using

J ≡ *λF.* G *λψ. F* (*λm.* (*ψ* · *r*)(*x, m*))

in the primality equation for *ϵL*, so

G *ϵ*(*x* ::: *L*) = G *λψ. ϵL*(*λm.* (*ψ* · *r*)(*x, m*)) def *ϵ*(*x* ::: *L*)

≡ J(*ϵL*) def J

= *ϵL* *λm.* J(*λψ. ψm*) *ϵL* prime (hypothesis)

= *ϵL* *λm.* G(*λψ.* (*ψ* · *r*)(*x, m*)) def J

= *λψ*'*. ϵL*(*λm.* (*ψ*' · *r*)(*x, m*)) *λm*'*.* G(*λψ. ψm*') (*λβ*)−1

= *ϵ*(*x* ::: *L*) *λm*'*.* G(*λψ. ψm*') def *ϵ*(*x* ::: *L*)

Hence *ϵL* = *λψ. ψ*(*eL*) for some unique map *e* : List*X* → *M* , which also satisfies *e* nil = *z* and *e*(*x* ::: *L*) = *r*(*x, eL*).

We can now define all sorts of operations on lists in the usual way. We shall need the following in particular.

Lemma 9.9 *n <* |*L*| iff E!*x*:*X.* (*n, x*) ∈ *L*, so we may define *L*@*n* by descrip- tion [[A](#_bookmark109), Section 9].

Lemma 9.10 [*L*]*φ* = 6*n <* |*L*|*. φ*(*n, L*@*n*) and ⟨*L*⟩*φ* = E*n <* |*L*|*. φ*(*n, L*@*n*).

Proof They satisfy the same recursion equations (*cf.* Remark [3.2](#_bookmark13)).

Definition 9.11 We define *concatenation* (++) on List*X* in the usual way by recursion:

nil ++ *L*2 = *L*2 and (*x* ::: *L*1) ++ *L*2 = *x* ::: (*L*1 ++ *L*2)*.*

Beware that ++ on List*X* is not the same as + on K(N × *X*) or ΩN×*X* .

Proposition 9.12 Using list induction we prove in the usual way that

(a) ++ is associative with unit nil ;

1. for any *f* : *X* → *Y* , List*f* is a homomorphism for nil and ++; and
2. List*X* is the free monoid on *X* with respect to ++.

Lemma 9.13 *x* = *L*@*m* for some *m* iff *L* = *L*' ++ {|*x*}| ++ *L*'' for some *L*', *L*'', where *L*' = take (*L, m*) and *L*'' = drop (*L, m* + 1) in functional programming

notation.

Lemma 9.14 List preserves equalisers.

Proof Let *X*) *i*) *Y f*) *Z* be an equaliser of overt discrete spaces. (This

*g*

exists, given *f* and *g*, because *Z* is discrete.) We show, using equational list induction on Γ ▶ *L* : List*Y* , that if Γ ▶ List*fL* = List*gL* : List*Z* then Γ ▶ E!*L*' :List*X. L* = List*iL*'. For the base case, nil *Y* = List*i* nil *X* . For the

induction step,

*fy* ::: List*fL* ≡ List*f* (*y* ::: *L*) = List*g*(*y* ::: *L*) ≡ *gy* ::: List*gL*

iff *fy* = *gy* Λ List*fL* = List*gL*. Hence if *L* = List*iL*' (by the induction hypothesis) and *y* = *ix* then *y* ::: *L* = List*i*(*x* ::: *L*').

Lemma 9.15 List preserves pullback over 1, *i.e.*

*X* + *Y*  ) *Y* List(*X* × *Y* ) ) *Y*

)

v

v ) v |−| )

|−|

v

*X* 1 List*X* N

Proof Define zip : List*X* × List*Y* → List(*X* × *Y* ) by

zip (*x* ::: *L*1*,y* ::: *L*2) = (*x, y*) ::: zip (*L*1*, L*2)

zip (*L*1*,* nil ) = nil = zip ( nil *, L*2)*.*

Now if Γ ▶ *L*1 : List*X* and Γ ▶

*i.e.* |*L*1| = |*L*2|, then *L*1 = List*π*0

*L*2 : List*Y*

make a commutative square,

and *L*2 = List*π*1

zip (*L*1*, L*2) ,

and Γ ▶ zip (*L*1*, L*2) : List(*X* × *Y* ) is the unique thing that does this.

zip (*L*1*, L*2)

Theorem 9.16 List preserves all finite connected limits.

Proof They may be obtained from equalisers and pullbacks like this.

# The free semilattice

Now that we have the free monoid, and [[C](#_bookmark106), Section 11] showed how to con- struct (stable effective) quotients of overt discrete objects by open equivalence

relations, the free semilattice exists. We want to show that this free semilat- tice is in fact K*X*. One way to do this would be to follow the motivation in Section [3](#_bookmark12), which we have now made legitimate. Instead, we shall identify the quotient map List*X* → K*X* for the semilattice laws directly.

Recall from Theorem [7.11](#_bookmark63) that K*π*1 is a homomorphism in the sense that

K*π*1 nil = nil and K*π*1(*x* ::: *L*) = *x* :: K*π*1*L*.

Lemma 10.1 K*π*1 is a homomorphism: K*π*1(*L*1 ++ *L*2) = K*π*1*L*1 + K*π*1*L*2.

Proposition 10.2 K*π*1 : List*X* → K*X* is an open surjection.

Proof It is an open map since both List*X* and K*X* are overt discrete [[C](#_bookmark106), Lemma 10.2]. For it to be an open surjection [[C](#_bookmark106), Definition 10.4] we have to show that Θ ≡ EK*π* T = T : ΣK*X*, where

1

*S* : K*X* ▶ Θ*S* ≡ EK*π*1 T*S* = E*L*:List*X.* (K*π*1*L* = *S*)*.*

Since K*π*1 nil = nil , we have Θ nil = T. Then

Θ*S* ≡ E*L.* (K*π*1*L* = *S*)

≤ E*L.* (K*π*1(*x* ::: *L*) = *x* :: *S*) above

≤ E*L*'*.* (K*π*1*L*' = *x* :: *S*) where *L*' = *x* ::: *L*

= Θ(*x* :: *S*) = S*x*Θ*S*

so Θ ≤ S*x*Θ. Hence Θ = *λS.* T by Corollary [4.12](#_bookmark34).

Lemma 10.3 K*π*1 is a natural transformation from List to K.

Proof Both composites in the square are of the form (*N, P* ) '→ (*N* ·*H, P* ·*H*), where *H* = Σ*π*1 · Σ*f* = Σid×*f* · Σ*π*1 .

Notation 10.4 Write *L*1*, L*2 : List*X* ▶ *L*1 ≈ *L*2 : Σ for the open con- gruence generated by the semilattice laws. Explicitly, this is of the form EL:List(List*X*)*.* (·· ·), where (·· ·) says that the list L of lists begins *L*1, ends *L*2 and its successive members are related by one of the semilattice laws. Being a congruence also means that if *L*1 ≈ *L*2 and *L*3 ≈ *L*4 then *L*1 ++*L*3 ≈ *L*2 ++*L*4.

Lemma 10.5 K*π*1 : List*X* → K*X* coequalises these laws, *i.e.* if *L*1 ≈ *L*2 then

K*π*1*L*1 = K*π*1*L*2.

Proof This is an equational induction over the list L, in which the induction step considers a single instance of a semilattice law *L*1 ≈ *L*2 in which two elements are either interchanged or coalesced. But K*π*1 is a homomorphism for +, and these laws hold in K*X*.

Lemma 10.6 K*π*1*L*1 ⊂ K*π*1*L*2 iff (6*n <* |*L*1|*.* E*m <* |*L*2|*. L*1@*n* = *L*2@*m*).

1

2

1

1

2

Proof

K*π L*

1

1

⊂ K*π L*

= [K*π L*] *λx.* ⟨K*π L* ⟩(*λy. x* = *y*) Notation [3.3](#_bookmark14)

= Σ2*π*1[*L*] *λx.* Σ2*π*1⟨*L*2⟩(*λy. x* = *y*) Remark [7.1](#_bookmark55)

= [*L*] *λnx.* ⟨*L*2⟩(*λmy. x* = *y*)

= (6*n <* |*L*1|*.* E*m <* |*L*2|*. L*1@*n* = *L*2@*m*)L. [9.10](#_bookmark81)

Lemma 10.7 If K*π*1*L*1 ⊂ K*π*1*L*2 then *L*1 ++ *L*2 ≈ *L*2.

Proof By equational list induction on *L*1. For the base case, nil ++ *L*2 = *L*2. Suppose that *x* ∈ K*π*1*L*. Then *x* = *L*@*m* for some *m <* |*L*|, so *L* = *L*++{|*x*|}++*L* for some lists *L* and *L* by Lemma [9.13](#_bookmark83), whence

(*x* ∈ K*π*1*L*) ≤ {|*x*|} ++ *L* ≈ *L* : Σ*.*

Hence for the induction step,

K*π*1(*x* ::: *L*1) ⊂ K*π*1*L*2 = (*x* ∈ K*π*1*L*2) Λ K*π*1*L*1 ⊂ K*π*1*L*2

≤ *x* ∈ K*π*1(*L*1 + *L*2) Λ *L*1 ++ *L*2 ≈ *L*2

≤ *x* ::: (*L*1 ++ *L*2) ≈ (*L*1 ++ *L*2) Λ *L*1 ++ *L*2 ≈ *L*2

≤ (*x* ::: *L*1) ++ *L*2 ≈ *L*2

Corollary 10.8 K*π*1*L*1 = K*π*1*L*2 iff *L*1 ≈ *L*1 ++ *L*2 ≈ *L*2, and so K*X* =

List*X/*≈. This coequaliser is valid in £ as well as in S [[C](#_bookmark106), Lemma 10.8].

Lemma 10.9 + : K*X* × K*X* → K*X* is well defined and is preserved by K*f* .

List*X* × List*X* K*π*1 × K*π*1)) K*X* × K*X*  ) Ω × Ω

*X* *X*

.

.

.

.

++ . +

.

.

.

v K*π*1 )

.v v

List*X*

K*X*) ) Ω*X*

Proof The rectangle commutes by Lemma [10.1](#_bookmark87), the top left map is a sur- jection by Proposition [10.2](#_bookmark88) and the bottom right is a Σ-split mono, so there is a unique fill-in. The map *f* : *X* → *Y* turns the diagram into a commuta-

tive cuboid, since K > Ω. K*π*1, + and ++ are natural by Theorem [7.11(b)](#_bookmark64),

Lemma [10.3](#_bookmark89), Lemma [7.4](#_bookmark57) and Proposition [9.12(b)](#_bookmark82).

Theorem 10.10 K*X* is the free semilattice (in the sense of +) on *X* in £.

List*X* K*π*1)) K*X*) ) Ω

{|(0*,* −)|}

ˆ...

.

.

...*ϵ*.

.

.

.

.

. *δ*

... .

.

.

. .v

*X*  ) *M*

Proof If *M* is an (imposed) semilattice then it is in particular a monoid, so by Proposition [9.12(c)](#_bookmark84) there is a unique homomorphism (for nil , ::: and ++) *ϵ* : List*X* → *M* . But as *M* also obeys the semilattice laws, *ϵ* factors through the coequaliser (Corollary [10.8](#_bookmark91)), giving the required mediator *δ* : K*X* → *M* .

This is a homomorphism (for nil and +) because K*π*1 is, and is surjective.

If *δ*' : K*X* → *M* is another homomorphism for nil and + then *δ*' · K*π*1 :

List*X* → *M* is a homomorphism for nil and ++, so *δ*' · K*π*1 = *δ* · K*π*1, and

*δ*' = *δ* because K*π*1 is surjective.

Proposition 10.11 Let Γ*, L* : K*X* ▶ *αL, βL* : *A* ≡ Σ*U* . Then, *cf.* Axiom [2.6](#_bookmark10) and Proposition [9.7](#_bookmark80),

Γ ▶ *α* nil = *β* nil Γ*, L* : K*X, x* : *X, αL* = *βL* ▶ *α*(*x* :: *L*) = *β*(*x* :: *L*) Γ*, L* : K*X* ▶ *αL* = *βL*

Proof Consider *α*¯ ≡ *α* · K*π*1*, β*¯ ≡ *β* · K*π*1 : List*X* → K*X* ⇒ *A*. These satisfy Proposition [9.7](#_bookmark80), so *α*¯ = *β*¯, whence *α* = *β* since K*π*1 is surjective.

Corollary 10.12 Any Σ-split subspace *U* ⊂ K*X* that contains nil and is closed under :: is *U* = K*X*.

Proof *U* can be expressed as the equaliser of some *α, β* : K*X* ⇒ *A*

[[B](#_bookmark110), Proposition 4.14].

# Overt compact subspaces

After this lengthy manipulation in the logical calculus of ASD, you may be left wondering what “admissible” or “modal” terms *L* : K*X* ever had to do with “finite subsets” or “compact open subspaces” of *X*. In fact, the topological results follow from the *modal* laws in Definition [3.4](#_bookmark15), and it is these that we assume of *L* : Ω, rather than admissibility (*L* : K*X*), which we deduce on the basis of Axiom [1.10](#_bookmark5). In view of the lack of a theory of parametric types for ASD, we first consider a *global* (non-parametric) modal element ▶ *L* : Ω.

Proposition 11.1 The open subspace *i* : *U* ⊂ *X* classified by *π* is overt (and discrete), with existential quantifier E*U* = ⟨*L*⟩ · E*i* and modal operator

⟨*L*⟩ = E*U* · Σ*i*. This justifies the notation ⟨*L*⟩*φ* ≡ E*x* ∈ *L. φx*, *cf.* the last

version of Lemma [6.2](#_bookmark48).

Proof The space *X* is overt by hypothesis, and *U* is an open subspace of it classified by *π*, so *U* is itself overt, with E*U* = E*X* · E*i*. What we have to show, therefore, is that

⟨*L*⟩· E*i* = E*X* · E*i* and ⟨*L*⟩· E*i* · Σ*i* = ⟨*L*⟩*.*

As usual, we regard an open predicate *θ* on the open subspace *U* ⊂ *X* (which is classified by *π*) as a predicate on *X* itself, with *θ* ≤ *π*. This means that we

represent E*i* by id and Σ*i* by *π* Λ (−). Using the last of the eight modal laws, the first equation is then

⟨*L*⟩(E*iθ*) = ⟨*L*⟩*θ* = E*x. πx* Λ *θx* = E*x. θx* = E*X* (E*iθ*)

and the second, for *φ* : Σ*X* , is ⟨*L*⟩(E*i* · Σ*iφ*) = ⟨*L*⟩(*π* Λ *φ*) = ⟨*L*⟩*φ*.

Proposition 11.2 The subspace *i* : *U* ⊂ *X* is also compact, with universal quantifier 6*U* = [*L*] · E*i* and modal operator [*L*] = 6*U* · Σ*i*. This justifies the notation [*L*]*φ* ≡ 6*x* ∈ *L. φx.*

Proof We have to show that Σ! E [*L*] · E*i* and [*L*] · E*i* · Σ*i* = [*L*]. Again we represent E*i* by id and Σ*i* by *π* Λ (−). Let *σ* : Σ and *u* : *U* , *i.e. u* : *X* with *πu* = T.

Suppose *σ* ≤ [*L*]*ψ*. Then *σ* ≤ [*L*]*ψ* Λ *πu* ≤ *ψu* by Remark [3.6](#_bookmark16), so (*λu*:*U. σ*) ≤ *ψ*.

Conversely, suppose *λu*:*U. σ* ≤ *ψ*, so *x* : *X* ▶ *σ* Λ *πx* ≤ *ψx*. Then

*σ* = *σ* Λ [*L*]*π* 7th modal law

≤ [*L*](*λx. σ* Λ *πx*) Euclid

≤ [*L*]*ψ* hypothesis

For the modal operator, with *φ* : Σ*X* , [*L*](E*i* · Σ*iφ*) = [*L*]*φ*.

The subspace *U* need not be Hausdorff, *i.e.* have decidable equality, as we haven’t assumed this of *X* itself.

Proposition 11.3 Conversely, let *U* ⊂ *X* be compact open. Then ▶ (*N, P* ) : Ω is modal, where *P* ≡ E*U* · Σ*i* and *N* ≡ 6*U* · Σ*i*.

Proof As for Proposition [11.10](#_bookmark97) below without *γ*.

Once again we have a result for *modal L* : Ω when we really want one for *admissible L* : K*X*, but now at last we are able to state, and so invoke, the Scott continuity Axiom [1.10](#_bookmark5).

Theorem 11.4 ▶ *L* : Ω*X* is admissible iff it is modal, and such terms corre- spond bijectively to compact open subspaces *U* ⊂ *X*. Moreover, a subspace *U* ⊂ *X* is compact open iff it is listable.

Proof Without loss of generality *U* = *X*. Applying Axiom [1.10](#_bookmark5) to *F* ≡ 6*X* ,

*αl* = T and *φl* = *πl*, where *X* is a compact overt discrete space, we have

▶ T = 6*X*T = E*l*:K*X.* 6*X* (*λx. x* ∈ *l*) = E*l*:K*X.* (6*x*:*X. x* ∈ *l*)*.*

This means that (6*X,* E*X* ) ⊂ *l*, where *l* is admissible and (6*X,* E*X*) is modal, whilst *l* ⊂ (6*X,* E*X*) since (6*X,* E*X*) is the greatest modal pair (Proposition [3.8](#_bookmark17)). Hence (6*X,* E*X* ) ~ *l*, which means that (6*X,* E*X* ) is admissible (and E*l* above is unique). Notice that the two universal quantifiers in

E*l*:K*X.* 6*y* ∈ *l.* E*u*:*U. x* = *iu* Λ 6*u*:*U.* E*x* ∈ *l. x* = *iu* : Σ are legitimate because both *U* and *l* define compact spaces.

In the last part, ⇐ follows from well known topological results, namely

that the image of a compact overt space is compact overt, and that any overt

subspace of a discrete space is open [[C](#_bookmark106)]. The more significant ⇒ comes from the fact that List*X* → K*X*.

Remark 11.5 We have to be careful with the notion of listability: we may legitimately use a listing to prove results about *U* only if the statements of those results does not depend on the *choice* of listing [[8](#_bookmark105), §6.6].

Definition 11.6 An object *X* that is listable in the above sense is called *Kuratowski finite*. If it is also Hausdorff then there is a listing without repetitions, in which case we say that *X* is simply *finite*. Recall from Propo-

sitions [3.8](#_bookmark17)–[3.9](#_bookmark18) that K*X* has a greatest element iff *X* is Kuratowski finite, and

is a lattice (in fact a Boolean algebra) iff *X* is finite.

Encouraged by this success, we use the same Axiom and proof for the

Theorem 11.7 Any Γ ▶ *L* : Ω is admissible iff it is modal.

Proof We have a directed union in the sense of Axiom [1.10](#_bookmark5),

*π* : Σ*X, x* : *X* ▶ *πx* = E*l*:K*X.* (6*y* ∈ *l. πy*) Λ (*x* ∈ *l*)*.*

This is preserved by any Γ ▶ *N* : ΣΣ*X* , so

Γ ▶ T = *N* (*λx. P* {*x*}) 7th modal law

= *N* E*l.* (6*y* ∈ *l. P* {*y*}) Λ (*λx. x* ∈ *l*) above

= E*l.* (6*y* ∈ *l. P* {*y*}) Λ *N* (*λx. x* ∈ *l*) Axiom [1.10](#_bookmark5)

= E*l.* *l* ⊂ (*N, P* ) Λ *N, P* ) ⊂ *l*) Notation [3.3](#_bookmark14)

= E*l*:K*X.* *l* = (*N, P* ) *,* Corollary [3.11](#_bookmark20)

where *l* is unique and admissible, so (*N, P* ) is itself admissible.

What are the results for parametric Γ ▶ *L* : K*X* corresponding to the compact open subspace *U* ⊂ *X* above? If we had a theory of parametric

types, in place of a single object *U* , we would have a *display map U* −−*d* Γ [[8](#_bookmark105), Chapter VIII]. The idea that each *Uγ* is overt compact is expressed by saying that *p* : *U* −−*d* Γ is an open proper map [[C](#_bookmark106), §7]. Also, where *U* ⊂ *X* was an open subspace, *U* ⊂ Γ × *X* is an open binary relation Γ *~7 X*. We call

this a *Kuratowski-finite subset of X dependent on* Γ.

Proposition 11.8 Let Γ ▶ (*N, P* ) : K*X*. Then there is a diagram as shown, in which the open subspace *i* : *U ‹*→ *X* ×Γ is classified by *λxγ. Pγ*{*x*} : Σ*X*×Γ, the three squares are pullbacks, and *p* is an open proper map.

*U*  ) (∈) ) 1

∩

v

∩

*i* T

v v

*p X* × Γ ) *X* × K*X*  ) Σ

*x,* (*N* '*,P* ') '→ *P* '{*x*}

*π*1 *π*1

v (*N, P* ) v

)

) Γ K*X*

Proof The composite *p* = *π*1 · *i* is open since *X* is overt. To show that *p* is proper, we must find the right adjoint of Σ*p* and verify the dual Frobenius law

[[C](#_bookmark106), Definition 7.3] and Beck–Chevalley condition. In fact, we shall also give

another formula for the left adjoint, and I claim that *E* · E*i* = EΓ

*X*

· E*i* = E*p* E

Σ*p* E *A* · E*i*.

Σ*U*

)

ˆ

T

E*i*

)

Σ*p*

*E* : *ψ* '→ *λγ. Pγ λx. ψ*(*x, γ*)

)

(( Σ*i*

Σ*X*×Γ

*A* : *ψ* '→ *λγ. Nγ* *λx. ψ*(*x, γ*)

) ΣΓ

First recall that (E*i* · Σ*iψ*)(*x, γ*) = *ψx* Λ (*x, γ*) ∈ *U* = *ψx* Λ *Pγ*{*x*}, and let

*x* : *X*, *γ* : Γ, *θ* : ΣΓ and *ψ* : Σ*X*×Γ.

For the left adjoint we require *E* · (E*i* · Σ*i*) = EΓ · (E*i* · Σ*i*).

*X*

For the unit of the right adjoint, idΣΓ ≤ *A* · (E*i* · Σ*i*) · Σ*π*1 . For the counit, (E*i* · Σ*i*) · (Σ*π*1 · *A*) · (E*i* · Σ*i*) ≤ (E*i* · Σ*i*).

For the dual Frobenius law, *A* · E*i*(*φ* V Σ*pθ*)*γ* = *A*(E*iφ*)*γ* V *θγ*.

Finally, we must show stability under pullback along *s* : ∆ → Γ. This follows by application of the same results, but for ∆ ▶ (*s*∗*N, s*∗*P* ) : K*X*.

Σ*s*∗*U* ( Σ*U s*∗*U*  ) *U*  ) (∈) ) 1

v v ∩

∩

v

∩

*i*

v

E*j* E*i*

v Σ*X*×*s* v

*j* T

*X* × *s* v v

Σ*X*×∆ ( Σ*X*×Γ

*X* × ∆ ) *X* × Γ ) *X* × K*X*  ) Σ

v v

v

*π*1

v

*A*' *A*

v Σ*s* v

*π*1

*s*

*π*1

(*N, P* ) v

Σ∆ ( ΣΓ

∆ ) Γ ) K*X*

The Beck–Chevalley condition for the top left square is that for the inverse image of an open inclusion [[C](#_bookmark106), Proposition 3.11]. In the the bottom left square, where

*A*'*θ* ≡ *λδ. N* (*sδ*) *λx. θ*(*x, δ*) *,*

the Beck–Chevalley condition is

Σ*s*(*Aψ*)*δ* = *N* (*sδ*) *λx. ψ*(*x, sδ*) = *A*'(Σ*X*×*sψ*)*δ.*

Corollary 11.9 In particular, (∈) *‹*→ *X* × K*X* → K*X* is open and proper.

Proposition 11.10 Conversely, let *i* : *U ‹*→ *X* × Γ be open such that *p* ≡

(*π*1 · *i*) : *R* → Γ is (open and) proper. Then *Nγφ* = 6*p*(*λu. φ*(*qu*)) and

*Pγφ* = E*p*(*λu. φ*(*qu*)) = E*x. φx* Λ (*x, γ*) ∈ *U* = E*x. φx* Λ *Pγ*{*x*}

a re modal ( where *q* ≡ (*π*0 · *i*) : *R* → *X*). We recover *U* from *P* as *Pγ*{*x*} =

.

(*x, γ*) ∈ *U*

Theorem 11.11 K*X* classifies Kuratowski finite subsets of *X*, in the sense that

open proper relations Γ *~7 X* as above

============================== Γ −→ K*X*

Theorem 11.12 K*X* classifies Kuratowski finite subsets of *X*.

*U*  ) (∈)

*p*

v

v

Γ ) K*X*

Indeed (∈) : *X ~7* K*X* is the generic Kuratowski-finite subset of *X* (dependent on K*X*). By this we mean that any pullback of it as shown is a Kuratowski-

finite subset of *X* dependent on Γ, and every Kuratowski-finite subset of *X*

dependent on Γ arises uniquely in this way.

# Acknowledgement

I would like to thank Peter Aczel, Andrej Bauer, Robin Cockett, Maria Emilia Maietti and Gavin Wraith for their helpful comments. This paper was ac- cepted for *Category Theory and Computer Science* 10, which was held in

Copenhagen in August 2004, and an abridged version appeared in the printed proceedings. However, I was unable to attend, as I had broken my leg two weeks earlier. Research on Abstract Stone Duality has been funded since September 2003 by the EPSRC project GR/S58522 of the same title, but the basic results of Sections [3](#_bookmark12)–[6](#_bookmark47) had been found in July–August 2003.

# References

1. Cockett, J. R., *List-arithmetic distributive categories: Locoi*, Journal of Pure and Applied Algebra 66 (1990), pp. 1–29.
2. Hofmann, M., “Extensional concepts in intensional type theory,” Ph.D. thesis, University of Edinburgh (1995).

URL <http://www.lfcs.inf.ed.ac.uk/reports/95/ECS-LFCS-95-327>

1. Johnstone, P., “Stone Spaces,” Number 3 in Cambridge Studies in Advanced Mathematics, Cambridge University Press, 1982.
2. Johnstone, P. T., *Vietoris locales and localic semi-lattices*, in: R.-E. Hoffmann and K. H. Hoffmann, editors, *Continuous Lattices and their Applications*, Marcel Dekker, 1985 pp. 155– 180.
3. Lambek, J. and P. J. Scott, “Introduction to Higher Order Categorical Logic,” Cambridge University Press, 1986.
4. Maietti, M. E., *Joyal’s arithmetic universes via type theory*, Electronic Notes in Theoretical Computer Science 69 (2003).
5. Maietti, M. E., *Reflection into models of ﬁnite decidable FP-sketches in an arithmetic universe*, Electronic Notes in Theoretical Computer Science 416 (2005).
6. Taylor, P., “Practical Foundations of Mathematics,” Cambridge University Press, 1999.
7. Vickers, S. J., “Topology Via Logic,” Cambridge Tracts in Theoretical Computer Science 5, Cambridge University Press, 1988.
8. Winskel, G., *Powerdomains and modality*, in: M. Karpinski, editor, *Foundations of Computation Theory*, number 158 in Lecture Notes in Computer Science (1983), pp. 505–514.

My papers on abstract Stone duality may be obtained from

[www.cs.man.ac.uk/∼pt/ASD](http://www.cs.man.ac.uk/~pt/ASD/index.dvi)

1. [Sober spaces and continuations](http://www.cs.man.ac.uk/~pt/ASD/sobsc.dvi). *Theory and Applications of Categories*, 10(12):248–299, 2002.
2. [Subspaces in abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/subasd.dvi). *Theory and Applications of Categories*, 10(13):300–366, 2002.
3. [Geometric and higher order logic using abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/geohol.dvi). *Theory and Applications of* *Categories*, 7(15):284–338, 2000.
4. [Non-Artin gluing in recursion theory and lifting in abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/nonagr.dvi). 2000.
5. [Inside every model of Abstract Stone Duality lies an Arithmetic Universe](http://www.cs.man.ac.uk/~pt/ASD/insema.dvi). *Category Theory and Computer Science 10*, Copenhagen, 2004.
6. [Scott domains in abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/pcfasd.dvi). March 2002.

[G–] [Local compactness and the Baire category theorem in abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/loccbc.dvi).

[*Electronic Notes in Theoretical Computer Science* 69](http://www.elsevier.nl/gej-ng/31/29/23/131/23/show/Products/notes/), Elsevier, 2003.

1. [Computably based locally compact spaces](http://www.cs.man.ac.uk/~pt/ASD/comblc.dvi). May 2003.

[H-] [An elementary theory of the category of locally compact locales](http://www.cs.man.ac.uk/~pt/ASD/undset-.dvi). APPSEM Workshop, Not- tingham, March 2003.

1. [An elementary theory of various categories of spaces and locales](http://www.cs.man.ac.uk/~pt/ASD/undset.dvi). November 2004.
2. [The Dedekind reals in abstract Stone duality](http://www.cs.man.ac.uk/~pt/ASD/reals.dvi). December 2004.
3. [A λ-calculus for real analysis](http://www.cs.man.ac.uk/~pt/ASD/ivt.dvi). December 2004.