Complexity Doctrines

James R. Otto, Jr.
Department of Mathematics and Statistics
McGill University, Montreal

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

June 13, 1995

ISBN 0-612-08143-5

Contents

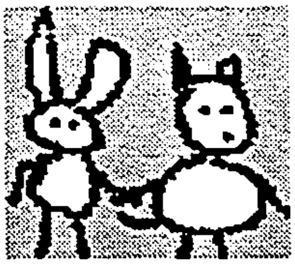
Résumé

On caractérise diverses classes de complexité comme des images dans $\mathbf{set}^2$, $\mathbf{set}^V$, et $\mathbf{set}^3$ de catégories initiales dans des doctrines de complexité. (Une doctrine est constituée des modèles d'une théorie de théories.) On caractérise de cette façon les fonctions de temps linéaire, d'espace polynômial, de temps polynômial, élémentaires dans le sens de Kalmar et les relations de hiérarchie de temps linéaire. (Notre modèle de machine sera les machines de Turing à plusieurs bandes, avec un nombre constant de bandes.) Ces doctrines étendent, en utilisant des compréhensions, les doctrines de premier ordre $\mathfrak{SM}$ et $\mathfrak{FP}$. On montre, en utilisant des diagrammes de produit dépendant, comment on peut étendre de cette façon la doctrine d'ordre supérieure $\mathfrak{LCC}$. D'autre part, en utilisant les numéraux de Church, on démontre que les compréhensions LCC résultantes n'apportent pas assez de contrôle sur les types d'ordre supérieur pour caractériser des classes de complexité. On montre aussi comment utiliser les esquisses et l'orthogonalité pour la spécification presque équationnelle.

Abstract

We characterize various complexity classes as the images in $\mathbf{set}^2$, $\mathbf{set}^V$, and $\mathbf{set}^3$ of categories initial in various complexity doctrines. (A doctrine consists of the models of a theory of theories.) We so characterize the linear time, P space, linear space, P time, and Kalmar elementary functions as well as the linear time hierarchy relations. (Our machine model is multi-tape Turning machines with constant numbers of tapes.) These doctrines extend, using comprehensions, the first order doctrines $\mathfrak{SM}$ and $\mathfrak{FP}$. We show, using dependent product diagrams, how to so extend the higher order doctrine $\mathfrak{LCC}$. However, using Church numerals, we show that the resulting LCC comprehensions do not provide enough control over higher order types to characterize complexity classes. We also show how to use sketches and orthogonality for almost equational specification.

Introduction



A doctrine consists of the models of a theory of theories. We do not directly repeat the old horror story of the student who proves many marvelous theorems about a theory with no models, as the theories in our complexity doctrines trivially have as models functor categories such as set^2, set^V, and set^3. (Here V is the partial order $\rightarrow \leftarrow$.) However, set^2, set^V, and set^3, while reasonable from a Newtonian/Platonic point of view, are, except for their low ends, much too big to easily fit into a physics limited by the speed of light (special relativity), hydrogen atoms (quantum theory), and round off error (chaos). So we use the images of initial categories in complexity doctrines to characterize fairly physical low ends of set^2, set^V, and set^3. (Some theories are categories.) That is, we so characterize the linear time, P space, linear space, P time, and Kalmar elementary functions as well as the linear time hierarchy relations, all on multi-tape Turing machines with constant numbers of tapes.

These complexity doctrines extend, using comprehensions, the SM (= symmetric monoidal) and the FP (= finite products) doctrines. We show, using dependent product diagrams, how to so extend the LCC (= locally cartesian closed) doctrine. However, using Church numerals, we show that the resulting LCC comprehensions do not provide enough control over higher order types to characterize complexity classes. (We eventually hope to overcome this using a combination of comprehensions and fibrations.) Along the way, we view sketches as certain presheaves, and show how to use sketches and orthogonality for almost equational specification.

This thesis is organized as four chapters/papers, each with its own introduction. We invite the reader to consider these introductions. (Chapter 2 is a variant of a proceedings paper.) Now we indicate the originality of this thesis.

The other work on categorical characterizations of complexity classes that we know of is [Huw76, Huw82]. We differ from it by using initial categories and gluing as in [Rom89, PR89, LS86], by using comprehensions, and by characterizing linear time.

Comprehensions descend from [Pav90, JMS91, Law70]. We free them from fibrations and relate them to the partial orders 2, V, and 3. We use comprehensions to understand tiers [BC92, Lei94, LM92] and to restrict the internal initiality of (base 1 and 2) NNO (= natural numbers objects): the partial orders 2, V, and 3 indicate how 'for loops' are allowed to nest. NNO [LS86, BW90, CRCM80], as well as comprehensions, are due to F. Lawvere.

The linear space, P time, and Kalmar elementary characterizations descend from [Bel92, Rit63, BC92, Cob65, LM92], but differ by using categories and (2- and 3-) comprehensions. We also distinguish between safe recursion and dependent safe recursion. (See the introduction to Chapter 4.) The P space characterization descends from [Tho72, Huw76], but differs by using V-comprehensions. While the P space characterization here pumps up linear space, that in [LM95] pumps up P time.

The linear time characterization descends from [Blo92, LM92, Lei94], but differs from [Blo92] by not having diagonal at tier 0, and from [Lei94] by not allowing machine registers to be copied in unit time. (See the introduction to Chapter 1.) We see the distinction between the SM and the FP doctrine as leading to the distinction between very safe recursion (for linear time) and safe recursion (for the other function classes), with diagonal needed to read the parameters more than once in safe recursion.

The characterization of the linear time hierarchy relations descends from [Wra78], but differs by using tiers rather than explicit bounds. There are similar characterizations of the P time hierarchy functions in [Bel92], and of the NP and the N linear space functions in [Lei94].

Dependent product diagrams (but not dp stacking) appear independently in [Ndj92]. (Our dp stacking proposition is from 1991.) Our use of sketches and orthogonality is our understanding of the use of sketches and injectivity in [Mak94]. (For the complex history of sketches see [Mak94, AR94, BW90, MP89].)

I thank my wife H. Tan, my parents J. Otto and R. Otto, my advisor M. Barr, my latest teacher M. Makkai, the tiers pioneers S. Bellantoni, S. Bloch, and D. Leivant, my teachers M. Bunge, R. Davis, D. Jurca, J. Lambek, C. Moore, W. Nico, and D. Thérien, as well as A. Blass, R. Blute, W. Boshuck, R. Cockett, D. Čubrić, V. Harnik, J. Loveys, F. Magnan, R. Paré, R. Squire, R. Seely, C. Wells, and A. Zappitelli for their help.

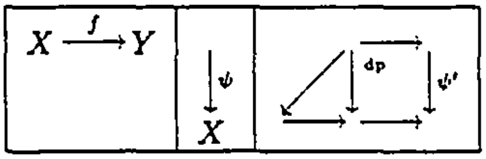
We have used xy-pic, latex2ϵ, and auc-tex.

Chapter 1

Tensor and Linear Time

Introduction

One might wonder, for example, how to cleanly combine functional and logic programming. Categorical logic may help answer such questions. Both functions $f : X \to Y$ and relations (or types) ψ on X are maps, and higher order types ψ' are from dependent product diagrams (Chapter 3), or more generally, from fibrations [BW90, Mak93, Bor94, Pav90]:



Thus categorical semantics of programming languages is an industry [P$^+$91].

After seeing a draft of [BC92], we soon realized that the Bellantoni-Cook composition is the serial composition in (the category which is) the Kripke structure on the partial order 2 (Section 1.3.1).

In categories having higher order types, recursion is defined by natural numbers objects (NNO) [LS86, BW90, CRCM80]. Categories having higher order types and NNO characterize Gōdel's system T (Section 1.2.4). Without the higher order types, but with a compatibility with tensor or product which would follow from having higher order types, categories with NNO and tensor or product characterize the primitive recursive functions [Rom89, PR89]. Indeed, the image in **set**, the category of small sets, of a category initial among such categories, is the set of primitive recursive functions.

H. Huwig uses bounded recursive characterizations of complexity classes and fragments of base 1 and 2 NNO to describe the corresponding subcategories of **set** [Huw76], and uses tensor rather than product to obtain an interesting model of such systems [Huw82]. Our work on complexity starts from [Rom89, PR89] and from tiered characterizations of complexity classes [BC92, Bel92, Blo92, Lei94].

We use tensor rather than product. But rather than view tensor as a broken product, we view it as a parallel composition, even though we implement it sequentially. Thus we can have both serial and parallel compositions of programs:

$f_1 \circ f_0$	$f_0 \otimes f_1$
$\xleftarrow[f_1]{} \ \xleftarrow[f_0]{}$	$\xleftarrow[f_0]{}$ $\xleftarrow[f_1]{}$

We formalize these combined compositions as symmetric monoidal (SM) categories (Section 1.2).

We understand tiers in terms of comprehensions [Pav90, JMS91, Law70] (Section 1.3). In particular, we abstract SM 2-comprehensions from the Kripke structure over the partial order 2. SM 2-comprehensions consist of modalities (or endo-functors) T, G and coercions (or natural transformations) η, ϵ, where T erases tier 0 inputs and outputs, G boosts them to tier 1, η forces safety, and ϵ coerces tier 1 data to tier 0. We use T and $\otimes$ to restrict the internal initiality of NNO to Leivant's flat and Bloch's very safe recursions. Then the image, in the Kripke structure over the partial order 2, of a category initial among SM categories having SM 2-comprehensions, tiers, and base 2 flat and very safe recursions, consists of the linear time functions on deterministic multi-tape Turing machines with constant numbers of tapes (Section 1.4).

Replacing tensor by product in our characterization of the linear time functions on multi-tape Turing machines recovers D. Leivant's characterization of the linear time functions on his register machines [Lei94]. These machines allow registers to be copied in unit time. In [Blo92] (from which our use of very safe recursion starts), there is an attempt to characterize the linear time functions on multi-tape Turing machines by what is essentially D. Leivant's characterization. We suspect that, in [Blo92], vector iteration is not fully considered. In particular, there is the 'diagonal issue' (Section 1.4.7).

There is the general heuristic that tier 0 operations such as quantifications and minimizations are automatically bounded. Indeed, [Bel92] so characterizes the P time hierarchy functions, and [Lei94] so characterizes the NP and N linear space functions. We so characterize the linear time hierarchy relations (Section 1.5), thus improving on [Wra78]. By the way, on multi-tape Turing machines, the linear time relations are not the N linear time relations [P+83, BDG90], and the linear time hierarchy relations are the bounded arithmetic relations [HP93, Woo86].

We construct initial categories using almost equational specification based on two layers of restricted equational specification: sketches and orthogonality [Mak94, AR94, Bor94] (Section 1.1, Appendices 1.A, 1.B).

We have attempted (in this chapter) to be largely accessible to nonspecialists. Thus we have pushed technical details to appendices. For background information and further details we suggest [BW90, Bor94, LS86, AR94, BW85].

1.1 Almost Equational Specification

1.1.1 Sketches

We modify [Mak94]. A *sketch theory* is (or see Appendix 1.A) an equational specification with a function height : sorts → N from sorts to natural numbers, such that

1. Operators (= function symbols) have arity 1. In particular, there are no constants.

2. Operators go to sorts of lower height. I.e. given an operator f to sort X' from sort X, which we write as

$$f\ x : X'\ [x : X]$$

we have height $X' <$ height X.

3. Only finitely many operators come from (as in 2.) any one sort.

Suppose that S is a sketch theory. Then an S *sketch* is a model (in set, as in Appendix 1.A) of S. Thus an S sketch s has

1. for each S sort X, a set $s\ X$,

2. for each S operator $f\ x : X'\ [x : X]$, a function $s\ f : s\ X \rightarrow s\ X'$,

such that, for each S equation $t_0 = t_1 : X'\ [x : X]$, the functions $s\ t_0,\ s\ t_1 : s\ X \rightarrow s\ X'$ are equal, where s interprets terms (i.e. strings of S operators applied to S sorted variables) t by $s\ x = \mathrm{id}$, $s\ (f\ t) = (s\ f) \circ (s\ t)$.

An S *homomorphism* $h : s \rightarrow s'$ is a map between models, i.e. h consists of, for each S sort X, a function $h\ X : s\ X \rightarrow s'\ X$ such that, for each S operator $f\ x : X'\ [x : X]$, the diagram

$$\begin{tikzcd}
s\ X \arrow[r, "h\ X"] \arrow[d, "s\ f"'] & s'\ X \arrow[d, "s'\ f"] \\
s\ X' \arrow[r, "h\ X'"'] & s'\ X'
\end{tikzcd}$$

commutes.

An S sketch s is *finite* iff the disjoint union $\sum_{\mathsf{S}\ \mathrm{sort}\ X} s\ X$ is finite.

We can specify any S sketch s by

1. taking enough *parameters* $x \in s\ X$, for S sorts X, so that all such are obtained by applying S operators,

2. taking enough equations $t_0 = t_1 : X'$ true in s to imply the rest, where the t_i are S terms with the variables replaced by parameters and evaluation in s is by $s\ x = x$, $s\ (f\ t) = (s\ f) \circ (s\ t)$.

We write this as the *context*

$$[\ldots\ t_0 = t_1 : X' \ \ldots\ x : X\ \ldots]$$

s is then, up to isomorphism, the initial model (in set) of the equational specification extending S with constants $x : X$ and equations $t_0 = t_1 : X'$. Thus an S homomorphism $h : s \to s'$ amounts to assigning parameters $x : X$ to $h\ x \in s'\ X$ in such a way that the equations $t_0 = t_1 : X'$ are true in s' under the evaluation $s'\ x = h\ x$, $s'\ (f\ t) = (s'\ f) \circ (s'\ t)$.

1.1.2 Orthogonality

Suppose that S is a sketch theory and that M is a set of S homomorphisms. An S sketch s is *orthogonal* to M iff S homomorphisms to s extend uniquely along $m \in M$, i.e. iff $\forall$

$$\begin{tikzcd} s & \\ {} \arrow[u, "h"] \arrow[r, "m"'] & {} \end{tikzcd}$$

with $m \in M$, $\exists!$ commuting

$$\begin{tikzcd} s & \\ {} \arrow[u, "h"] \arrow[r, "m"'] & {} \arrow[ul, "\tilde{h}"'] \end{tikzcd}$$

(Orthogonality is a restricted form of equational specification as it, given enough colimits, induces idempotent triples.)

A *basic almost equational specification* (S, M) consists of

1. a sketch theory S,

2. a set M of S homomorphisms between finite S sketches.

The *models* of (S, M) are the S sketches orthogonal to M, and the maps between them are the S homomorphisms between them.

As an example, we begin to specify serial composition (following [Mak94]). S has sorts (where $\rightsquigarrow$ is our comment symbol)

$$\begin{array}{ll} C_0 & \rightsquigarrow \text{objects} \\ C_1 & \rightsquigarrow \text{maps} \\ C_2 & \rightsquigarrow \text{triangles} \end{array}$$

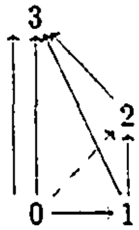
and operators

$$d_i\, x : C_j\; [x : C_{j+1}] \qquad \text{for } 0 \le j \le 1,\ 0 \le i \le j+1$$

(The face operator d_i omits vertex i.) Finally, S has equations

$$d_j\, d_i\, x = d_i\, d_{j+1}\, x : C_0\; [x : C_2] \qquad \text{for } 0 \le i \le j \le 1$$

As the triangles will be the graph of a serial composition partial function (namely $\tilde{\circ}$ below), we wish to specify associativity. For this we use the tetrahedron



with faces x_i (which omit vertex i). The associativity is that $d_1\, x_1 = d_1\, x_2$ if one has the conjunction of $d_j\, x_i = d_i\, x_{j+1}$ for $0 \le i \le j \le 2$, $(i,\ j) \ne (1,\ 1)$. We write this as the assertion

$$\begin{array}{l} \{d_1\, x_1 = d_1\, x_2 : C_1 \\ \;[d_0\, x_0 = d_0\, x_1 : C_1 \\ \;\;d_1\, x_0 = d_0\, x_2 : C_1 \\ \;\;d_2\, x_0 = d_0\, x_3 : C_1 \\ \;\;d_2\, x_1 = d_1\, x_3 : C_1 \\ \;\;d_2\, x_2 = d_2\, x_3 : C_1 \\ \;\;x_0 : C_2 \quad x_1 : C_2 \quad x_2 : C_2 \quad x_3 : C_2]\} \end{array}$$

An S sketch s models this assertion precisely when it is orthogonal to the S homomorphism

$$\begin{tikzcd}
\begin{array}{l}
[d_0\ x_0 = d_0\ x_1 : C_1 \\
\ d_1\ x_0 = d_0\ x_2 : C_1 \\
\ d_2\ x_0 = d_0\ x_3 : C_1 \\
\ d_2\ x_1 = d_1\ x_3 : C_1 \\
\ d_2\ x_2 = d_2\ x_3 : C_1 \\
\ x_0 : C_2 \quad x_1 : C_2 \quad x_2 : C_2 \quad x_3 : C_2]
\end{array} \arrow[d, "m_0"] \\
\begin{array}{l}
[d_1\ x_1 = d_1\ x_2 : C_1 \\
\ d_0\ x_0 = d_0\ x_1 : C_1 \\
\ d_1\ x_0 = d_0\ x_2 : C_1 \\
\ d_2\ x_0 = d_0\ x_3 : C_1 \\
\ d_2\ x_1 = d_1\ x_3 : C_1 \\
\ d_2\ x_2 = d_2\ x_3 : C_1 \\
\ x_0 : C_2 \quad x_1 : C_2 \quad x_2 : C_2 \quad x_3 : C_2]
\end{array}
\end{tikzcd}$$

where $m_0\ x_i = x_i$. Indeed, the assertion is just notation for the homomorphism.

Further, we wish to specify that the triangles are the graph of a serial composition partial function. Given

$$\xleftarrow{f_1}\ \xleftarrow{f_0}$$

we want a unique triangle $f_1 \tilde{\circ} f_0 =$

$$\begin{tikzcd}
& {} \\
{} \arrow[r, "f_0"'] \arrow[ur] & {} \arrow[u, "f_1"']
\end{tikzcd}$$

We write this as the $\tilde{\circ}$ assertion

$$\begin{array}{l}
\{!\ f_1 \tilde{\circ} f_0 : C_2 \\
\quad d_0\ (f_1 \tilde{\circ} f_0) = f_1 : C_1 \quad d_2\ (f_1 \tilde{\circ} f_0) = f_0 : C_1 \\
\quad [d_1\ f_1 = d_0\ f_0 : C_0 \quad f_1 : C_1 \quad f_0 : C_1]\}
\end{array}$$

where the ! indicates uniqueness, namely that the following uniqueness assertion is implied.

$$\begin{array}{l}
\{x = x' : C_2 \\
\ [d_0\, x = f_1 : C_1 \quad d_0\, x' = f_1 : C_1 \\
\ \ d_2\, x = f_0 : C_1 \quad d_2\, x' = f_0 : C_1 \\
\ \ d_1\, f_1 = d_0\, f_0 : C_0 \quad x : C_2 \quad x' : C_2 \quad f_1 : C_1 \quad f_0 : C_1]\}
\end{array}$$

An S sketch s models the $\tilde{o}$ assertion iff s is orthogonal to the S homomorphism

$$\begin{tikzcd}
{[d_1\, f_1 = d_0\, f_0 : C_0 \quad f_1 : C_1 \quad f_0 : C_1]} \arrow[d, "m_1"] \\
{\begin{array}{l} [d_0\, x = f_1 : C_1 \quad d_2\, x = f_0 : C_1 \\ \ d_1\, f_1 = d_0\, f_0 : C_0 \quad x : C_2 \quad f_1 : C_1 \quad f_0 : C_1] \end{array}}
\end{tikzcd}$$

where $m_1\, f_i = f_i$. (The uniqueness assertion results from the transformation $m_1 \mapsto m_1^*$ of Appendix 1.B.)

1.1.3 Essentially Algebraic Specification

Basic almost equational specifications (Section 1.1.2) are painfully low level. We sugar them to a variant of Freyd's essentially algebraic specifications [FS90]. For example, we respecify the above fragment of serial composition by

$$\begin{array}{l}
C_0 \ \leadsto \ \text{objects} \\
C_1 \ \leadsto \ \text{maps} \\
d\, x : C_0\, [x : C_1] \ \leadsto \ \text{domain (was } d_1) \\
c\, x : C_0\, [x : C_1] \ \leadsto \ \text{codomain (was } d_0) \\
\{f_1 \circ f_0 : C_1 \\
\ \ d\,(f_1 \circ f_0) = d\, f_0 : C_0 \quad c\,(f_1 \circ f_0) = c\, f_1 : C_0 \\
\ \ [d\, f_1 = c\, f_0 : C_0 \quad f_1 : C_1 \quad f_0 : C_1]\} \\
\{(f_2 \circ f_1) \circ f_0 = f_2 \circ (f_1 \circ f_0) : C_1 \\
\ \ [f_2 : C_1 \quad f_1 : C_1 \quad f_0 : C_1]\}
\end{array}$$

This last assertion has, from the occurrences of $\circ$, the implied conditions

$$\begin{aligned}
&d\,(f_2 \circ f_1) = c\ f_0 : C_0\\
&d\ f_2 = c\ f_1 : C_0\\
&d\ f_2 = c\,(f_1 \circ f_0) : C_0\\
&d\ f_1 = c\ f_0 : C_0
\end{aligned}$$

We recover the previous basic specification by replacing the conditional operator

$$\begin{tikzcd}
& {} \arrow[dl, dashed, "\circ"'] \arrow[dr] & \\
C_1 & & C_1 \times C_1
\end{tikzcd}$$

by the sort and operators

$$\begin{tikzcd}
& C_2 \arrow[dl, "d_1"'] \arrow[d, "d_0"] \arrow[dr, "d_2"] & \\
C_1 & C_1 & C_1
\end{tikzcd}$$

and by unnesting $(f_2 \circ f_1) \circ f_1 = f_2 \circ (f_1 \circ f_0)$ to $d_1\ x_1 = d_1\ x_2$ if one has the conjunction of

$$\begin{aligned}
&x_0 = f_2 \,\bar{\circ}\, f_1\\
&x_1 = (d_1\ x_0) \,\bar{\circ}\, f_0\\
&x_3 = f_1 \,\bar{\circ}\, f_0\\
&x_2 = f_2 \,\bar{\circ}\, (d_1\ x_3)
\end{aligned}$$

Similarly, given suitable layers of conditional operator (= function symbol) assertions and conditional equation assertions on top of a sketch theory, we can inductively unsugar to the basic form (S, M) by

1. using graphs of conditional operators,

2. unnesting [Hod93] equations.

Given the conditional operator assertion

$$\begin{aligned}
&\{\ldots\ f\ x_0\ x_1\ \ldots\ x_{n-1} : X\ \ldots\\
&\ [\ldots\ t_0 = t_1 : X'\ \ldots\ x_i : X_i\ \ldots]\}
\end{aligned}$$

where the context has already been unsugared, add the sort and operators

$$\begin{tikzcd}
& f \arrow[dl, "d"'] \arrow[d, "d_0"] \arrow[dr, "d_1"] \arrow[drr, phantom, "\cdots"] \arrow[drrr, "d_{n-1}"] & & & \\
X & X_0 & X_1 & \cdots & X_{n-1}
\end{tikzcd}$$

to S and then unsugar the conditional operator assertion to assertions

$$\begin{aligned}
&\{!\ \ldots\ \tilde{f}\, x_0\, x_1\ \ldots\ x_{n-1} : f\ \ldots\ d_i\, \tilde{f}\, x_0\, x_1\ \ldots\ x_{n-1} = x_i : X_i\ \ldots \\
&\quad [\ldots\ t_0 = t_1 : X'\ \ldots\ x_i : X_i\ \ldots]\} \\
&\{\ldots\ t_0 = t_1 : X'\ \ldots \\
&\ \ [\ldots\ d_i\, x = x_i : X_i\ \ldots\ x : f\ \ldots\ x_i : X_i\ \ldots]\}
\end{aligned}$$

After unnesting (which introduces variables for subterms)

$$[\ldots\ y = f\, x_0\, x_1\ \ldots\ x_{n-1} : X\ \ldots]$$

can be unsugared to

$$[\ldots\ y = d\, x : X\ \ldots\ d_i\, x = x_i : X_i\ \ldots\ x : f\ \ldots]$$

(Sometimes there exist more efficient unsugarings having equivalent categories of models.)

1.2 Tensor and System T

1.2.1 Serial Composition

We finish specifying serial composition by adding (to the specification of Section 1.1.3) the identity maps.

$$\begin{aligned}
&\{\mathrm{id}\ X : C_1 \quad d\ \mathrm{id}\ X = X : C_0 \quad c\ \mathrm{id}\ X = X : C_0\, [X : C_0]\} \\
&\{f \circ \mathrm{id}\ d\, f = f : C_1 \quad (\mathrm{id}\ c\, f) \circ f = f : C_1\, [f : C_1]\}
\end{aligned}$$

Models of this specification are called *categories*. E.g. set is the category of (small, for a convenient Grothendieck universe) sets and functions.

1.2.2 Parallel Composition

As we said in the introduction, the tensor is parallel composition. In order to have examples such as vector spaces (Appendix 1.D), we abstract combined serial and parallel composition ($\circ$ and $\otimes$) as *symmetric monoidal (= SM)* categories, which we almost equationally specify following [Tro92].

An *SM category* is (or see the specification below) a category **C** together with *tensor* and *unit* functors

$$\otimes : C \times C \rightarrow C \qquad \top : 1 \rightarrow C$$

as well as *associativity*, *symmetry*, and *left identity* natural isomorphisms

$$\begin{array}{c} \alpha\, X\ Y\ Z : X \otimes (Y \otimes Z) \rightarrow (X \otimes Y) \otimes Z \\ \sigma\, X\ Y : X \otimes Y \rightarrow Y \otimes X \\ \lambda\, X : \top \otimes X \rightarrow X \end{array}$$

satisfying

$$(\sigma\, Y\ X) \circ (\sigma\, X\ Y) = \mathrm{id} \qquad \sigma \top \top = \mathrm{id}$$

as well as the pentagon, triangle, and hexagon coherence conditions of Appendix 1.C. E.g. set (Section 1.2.1), with $\top = \{0\}$ and $\otimes = \times$ (where $X \times Y = \{(x,\ y) \mid x \in X,\ y \in Y\}$), is an SM category. (Actually, for SM categories, but not for SMC categories (Section 1.2.4), there is no loss of generality in taking the α's and λ's to be identities [JS91].)

As the special cases $f \otimes Y = f \otimes \mathrm{id}\, Y$, $X \otimes g = (\mathrm{id}\, X) \otimes g$ suffice, we leave the general case $f \otimes g$ implicit. We also use that, given the hexagon condition and that σ is natural, for α to be natural it is enough that $\alpha\ X\ Y\ Z$ be natural in X. (Apply the hexagon twice.) Thus SM categories are specified by the categories assertions of Sections 1.1.3, 1.2.1, the coherence assertions of Appendix 1.C, and the following. (We omit empty contexts [ ].)

$\rightsquigarrow$ Tensor and unit

$$\begin{array}{l} \{X \otimes Y : C_0\ [X : C_0 \quad Y : C_0]\} \\ \{f \otimes Y : C_1 \quad Y \otimes f : C_1 \\ \ \ d\,(f \otimes Y) = (d\, f) \otimes Y : C_0 \quad c\,(f \otimes Y) = (c\, f) \otimes Y : C_0 \\ \ \ d\,(Y \otimes f) = Y \otimes (d\, f) : C_0 \quad c\,(Y \otimes f) = Y \otimes (c\, f) : C_0 \\ \ \ [f : C_1 \quad Y : C_0]\} \end{array}$$

$\{\top : C_0\}$

$\rightsquigarrow$ Associativity, symmetry, and left unit

$\{\alpha\, X\, Y\, Z : C_1 \quad \alpha_1\, X\, Y\, Z : C_1 \quad c\, \alpha\, X\, Y\, Z = (X \otimes Y) \otimes Z : C_0$

$\ d\, \alpha_1\, X\, Y\, Z = (X \otimes Y) \otimes Z : C_0$

$\ (\alpha_1\, X\, Y\, Z) \circ (\alpha\, X\, Y\, Z) = \mathrm{id}\,(X \otimes (Y \otimes Z)) : C_1$

$\ (\alpha\, X\, Y\, Z) \circ (\alpha_1\, X\, Y\, Z) = \mathrm{id}\,((X \otimes Y) \otimes Z) : C_1$

$\ [X : C_0 \quad Y : C_0 \quad Z : C_0]\}$

$\{\sigma\, X\, Y : C_1$

$\ d\, \sigma\, X\, Y = X \otimes Y : C_0 \quad c\, \sigma\, X\, Y = Y \otimes X : C_0$

$\ (\sigma\, Y\, X) \circ (\sigma\, X\, Y) = \mathrm{id}\,(X \otimes Y) : C_1$

$\ [X : C_0 \quad Y : C_0]\}$

$\{\sigma\, \top\, \top = \mathrm{id}\,(\top \otimes \top)\}$

$\{\lambda\, X : C_1 \quad \lambda_1\, X : C_1$

$\ c\, \lambda\, X = X : C_0 \quad d\, \lambda_1\, X = X : C_0$

$\ (\lambda_1\, X) \circ (\lambda\, X) = \mathrm{id}\,(\top \otimes X) : C_0$

$\ (\lambda\, X) \circ (\lambda_1\, X) = \mathrm{id}\, X : C_0$

$\ [X : C_0]\}$

$\rightsquigarrow$ Functorality

$\{((c\, f) \otimes g) \circ (f \otimes (d\, g)) = (f \otimes (c\, g)) \circ ((d\, f) \otimes g) : C_1$

$\ [f : C_1 \quad g : C_1]\}$

$\{(\mathrm{id}\, X) \otimes Y = \mathrm{id}\,(X \otimes Y) : C_1\ [X : C_0 \quad Y : C_0]\}$

$\{(f_1 \circ f_0) \otimes Y = (f_1 \otimes Y) \circ (f_0 \otimes Y) : C_1$

$\ [f_0 : C_1 \quad f_1 : C_1 \quad Y : C_0]\}$

$\rightsquigarrow$ Naturality

$\{(\alpha\, (c\, f)\, Y\, Z) \circ (f \otimes (Y \otimes Z)) = ((f \otimes Y) \otimes Z) \circ (\alpha\, (d\, f)\, Y\, Z) : C_1$

$\ [f : C_1 \quad Y : C_0 \quad Z : C_0]\}$

$\{(\sigma\, (c\, f)\, Y) \circ (f \otimes Y) = (Y \otimes f) \circ (\sigma\, (d\, f)\, Y) : C_1$

$\ [f : C_1 \quad Y : C_0]\}$

$\{(\lambda\, c\, f) \circ (\top \otimes f) = f \circ (\lambda\, d\, f) : C_1\ [f : C_1]\}$

1.2.3 Unary Numbers

We write N for the set of natural numbers $\{0, 1, 2, \ldots\}$. The initial model in set of *unary*

$$N \qquad \{0 : N\} \qquad \{s\, n : N\, [n : N]\}$$

is

$$\top \xrightarrow{0} N \xrightarrow{s} N$$

where $s\, n = n + 1$. The terms $s^n\, 0$ can be identified with the unary (= base 1) numerals.

1.2.4 System T

In set we have the abstraction (= Currying)

$$\frac{f : W \times X \to Y}{\Lambda\, f : W \to Y^X}$$

where $(\Lambda\, f)\, w = (x \mapsto f\, w\, x)$. In SM categories we abstract this to the linear implication

$$\frac{f : W \otimes X \to Y}{\Lambda\, f : W \to X \multimap Y}$$

SM categories having all linear implications are called *symmetric monoidal closed* (= SMC) categories and are specified in Appendix 1.D.

In an SMC category **C**, a *natural numbers object (= NNO)*

$$\top \xrightarrow{0} N \xrightarrow{s} N$$

is an initial model of unary in **C**, which is that, $\forall\, g : \top \to Y$, $h : Y \to Y$, $\exists!$ **C** commuting

$$\begin{tikzcd}
\top \arrow[r, "0"] \arrow[dr, "g"'] & N \arrow[r, "s"] \arrow[d, "f"] & N \arrow[d, "f"] \\
& Y \arrow[r, "h"'] & Y
\end{tikzcd}$$

We specify this by

$$\begin{array}{l} \{!\, R\,g\,h : C_1 \quad d\,R\,g\,h = N : C_0 \\ \quad (R\,g\,h) \circ 0 = g : C_1 \quad (R\,g\,h) \circ s = h \circ R\,g\,h : C_1 \\ \quad [d\,g = \top : C_0 \quad d\,h = c\,g : C_0 \quad c\,h = c\,g : C_0 \quad g : C_1 \quad h : C_1]\} \end{array}$$

where, roughly as in Section 1.1.2, ! indicates uniqueness.

This last assertion, together with those of Sections 1.1.3, 1.2.1, 1.2.2 and Appendices 1.C, 1.D specify the doctrine $\mathfrak{T}$ of SMC categories having NNO and witnessed structure. E.g. set (Section 1.2.1), with $\top = \{0\}$, $\otimes = \times$, $N = \{0,\ 1,\ 2,\ \ldots\}$, $0 = 0$, $s = (n \mapsto n + 1)$, is in $\mathfrak{T}$. By the arguments in Appendix 1.B, there exists an initial category $\mathbf{I}$ in $\mathfrak{T}$. Thus there is a unique $\mathfrak{T}$ functor $i : \mathbf{I} \to \mathbf{set}$. We also have the functor

$$\begin{array}{c} \Gamma = \mathbf{I}(\top,\ \_) : \mathbf{I} \to \mathbf{set} \\ X \mapsto \mathbf{I}(\top,\ X) = \{\mathbf{I} \text{ map } f \mid d\,f = \top,\ c\,f = X\} \\ f \mapsto f \circ \_ \end{array}$$

Proposition 1.2.4.1
For $\mathbf{I}$ *initial in* $\mathfrak{T}$

1. $\Gamma\,\top = \{\mathrm{id}\,\top\}$.

2. $\Gamma(X \otimes Y) = \{(x \otimes y) \circ (\lambda\,\top)^{-1} \mid x \in \Gamma\,X,\ y \in \Gamma\,Y\}$.

3. $\Gamma\,N = \{s^n\,0 \mid n \in N\}$.

4. *Even up to natural isomorphism,* Γ *is not a* $\mathfrak{T}$ *functor.*

5. *The functions* $N^I \to N^{I'}$ *in* set *of Gödel's system T [GLT89, Ros84] are precisely those of the form* $i\,f$ *for* $\mathbf{I}$ *maps* $f : N^{\otimes I} \to N^{\otimes I'}$.

Proof. 1.–3. See Appendix 1.F.

4. $\Gamma\,(N \multimap N)$ is countable while N^N is not.

5. $N^{\otimes I}$ is defined by $N^{\otimes 0} = \top$, $N^{\otimes (I+1)} = N \otimes N^{\otimes I}$. By 1.–3., i agrees, up to natural isomorphism, with Γ on $\mathbf{I}$ maps $f : N^{\otimes I} \to N^{\otimes I'}$. By Appendix 1.B

and (as in [PRS9])

$$\begin{tikzcd}
\top \arrow[r, "0"] \arrow[dr, "id"'] & N \arrow[r, "s"] \arrow[d, "\tau"] & N \arrow[d, "\tau"] \\
 & \top \arrow[r, "id"'] & \top
\end{tikzcd}$$

$$\begin{tikzcd}
\top \arrow[r, "0"] \arrow[d, "\lambda^{-1}"'] & N \arrow[r, "s"] \arrow[d, "\delta"] & N \arrow[d, "\delta"] \\
\top \otimes \top \arrow[r, "0\otimes 0"'] & N \otimes N \arrow[r, "s\otimes s"'] & N \otimes N
\end{tikzcd}$$

terminal maps τ and diagonal maps δ are definable in **I**. τ and δ convert the SMC structure to a CC (= cartesian closed) structure. Thus system T differs from *i* only in that T specifies just fragments of the uniqueness (the !'s) for R (above) and Λ (Appendix 1.D). □

A starting point for Bloch's very safe recursion (Section 1.4.1) is

Proposition 1.2.4.2
In an SMC category with NNO, $\forall\, g : X \to Y,\ h : Y \to Y,\ \exists!$ *commuting*

$$\begin{tikzcd}
\top \otimes X \arrow[r, "0\otimes X"] \arrow[d, "\lambda"'] & N \otimes X \arrow[r, "s\otimes X"] \arrow[d, "f"] & N \otimes X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y \arrow[r, "h"'] & Y
\end{tikzcd}$$

Proof. Consider

$$\top \xrightarrow{\Lambda(g\circ\lambda)} X \multimap Y \xrightarrow{X\multimap h} X \multimap Y$$

□

1.3 Comprehensions and Tiers

1.3.1 Comprehensions

We understand tiers in terms of comprehensions. (In Chapter 3 we begin to think about comprehensions in conjunction with higher order types.) Our starting point for this was recognizing that the Bellantoni-Cook composition [BC92] is the serial composition in the Kripke structure on the partial order 2. This Kripke structure is the cotensor 2 $\multimap$ set (Appendix 1.E), and has as objects the functions $X : X_0 \to X_1$, and as maps the set commuting squares

$$\begin{tikzcd} X_0 \arrow[r, "f_0"] \arrow[d, "X"'] & Y_0 \arrow[d, "Y"] \\ X_1 \arrow[r, "f_1"'] & Y_1 \end{tikzcd}$$

In 2 $\multimap$ set we have the 2 tiers of numbers

$$N_0 \quad = \quad \begin{tikzcd} N \arrow[d, "n\mapsto 0"] \\ \top \end{tikzcd} \qquad\qquad N_1 \quad = \quad \begin{tikzcd} N \arrow[d, "\mathrm{id}"] \\ N \end{tikzcd}$$

We first abstract 2 $\multimap$ set to 2 $\multimap$ **C**, with **C** an SM category and thus a 0-cell in the 2-category $\mathfrak{SM}$ (Appendix 1.E). Notice that the ordinal 2 is the partial order $0 \to 1$. Thus the endomorphisms end(2) of 2 form a partially ordered monoid. Reversing the multiplication, but not the partial order, set $\mathbf{M} = \mathrm{end}(2)^{\circ}$. Then the right action of **M** on 2 induces a left action of **M** on 2 $\multimap$ **C**, i.e. a 2-functor $\mathbf{M} \to \mathfrak{SM}$. Abstracting from 2 $\multimap$ **C**, *SM 2-comprehensions* are just (or see the specification below) 2-functors $\mathbf{M} \to \mathfrak{SM}$.

(In Chapters 4, 2, instead of starting from the partial order 2, we start from the partial order 3 to characterize the Kalmar elementary functions, and from the partial order $V = \to \leftarrow$ to characterize the P space functions.)

M has the elements T $(x \mapsto 1)$, id, G $(x \mapsto 0)$. We name the point-wise partial order $\epsilon : G \to \mathrm{id}$, $\eta : \mathrm{id} \to T$.

Proposition 1.3.1.1
M *has generators T, G, η, ϵ and relations*

$$\begin{array}{cccc} T\,G = G & \eta\,T = \mathrm{id} & T\,\eta = \mathrm{id} & G\,\eta = \eta \circ \epsilon \\ G\,T = T & \epsilon\,G = \mathrm{id} & T\,\epsilon = \eta \circ \epsilon & G\,\epsilon = \mathrm{id} \end{array}$$

as a 2-category.

Proof. E.g. $T^2 = T\ (G\ T) = (T\ G)\ T = G\ T = T$ and $\epsilon\ T = \epsilon\ (G\ T) = (\epsilon\ G)\ T = \mathrm{id}$. □

So an *SM 2-comprehension* consists of

1. an SM category $\mathbf{C}$,
2. functors (indeed, modalities) T, $G : \mathbf{C} \to \mathbf{C}$ preserving, up to identity, $\top$, $\otimes$, α, σ, λ and satisfying the relations of Proposition 1.3.1.1,
3. natural transformations (indeed, coercions) $\eta : \mathrm{id} \to T$, $\epsilon : G \to \mathrm{id}$ satisfying

$$\begin{array}{cc} \eta\ \top = \mathrm{id} & \epsilon\ \top = \mathrm{id} \\ \eta\ (X \otimes Y) = \eta\ X \otimes \eta\ Y & \epsilon\ (X \otimes Y) = \epsilon\ X \otimes \epsilon\ Y \end{array}$$

as well as the relations of Proposition 1.3.1.1.

Thus an SM 2-comprehension is specified by Sections 1.1.3, 1.2.1, 1.2.2, Appendix 1.C, and the following.

$\rightsquigarrow$ 1-cells

$\{T\ X : C_0\ [X : C_0]\}$

$\{T\ f : C_1 \quad d\ T\ f = T\ d\ f : C_0 \quad c\ T\ f = T\ c\ f : C_0\ [f : C_1]\}$

$\{T\ \mathrm{id}\ X = \mathrm{id}\ T\ X : C_0\ [X : C_0]\}$

$\{T\ (f_1 \circ f_0) = T\ f_1 \circ T\ f_0 : C_1\ [f_1 : C_1 \quad f_0 : C_1]\}$

$\{T\ \top = \top : C_0\}$

$\{T\ (f \otimes Y) = T\ f \otimes T\ Y : C_1\ [f : C_1 \quad Y : C_0]\}$

$\{T\ \alpha\ X\ Y\ Z = \alpha\ (T\ X)\ (T\ Y)\ (T\ Z) : C_1$
$[X : C_0 \quad Y : C_0 \quad Z : C_0]\}$

$\{T\ \sigma\ X\ Y = \sigma\ (T\ X)\ (T\ Y) : C_1\ [X : C_0 \quad Y : C_0]\}$

$\{T\ \lambda\ X = \lambda\ T\ X : C_1\ [X : C_0]\}$

... similar for G ...

$\rightsquigarrow$ 2-cells

$\{\eta\ X : C_1 \quad d\ \eta\ X = X : C_0 \quad c\ \eta\ X = T\ X : C_0\ [X : C_0]\}$

$\{(\eta\ c\ f) \circ f = (T\ f) \circ \eta\ d\ f : C_1\ [f : C_1]\}$

$\{\eta\ \top = \mathrm{id}\ \top : C_1\}$

$\{\eta\,(X \otimes Y) = \eta\, X \otimes \eta\, Y : C_1\ [X : C_0 \quad Y : C_0]\}$
$\{\epsilon\, X : C_1 \quad d\,\epsilon\, X = G\, X : C_0 \quad c\,\epsilon\, X = X : C_0\ [X : C_0]\}$
$\{(\epsilon\, c\, f) \circ G\, f = f \circ \epsilon\, d\, f : C_1\ [f : C_1]\}$
$\{\epsilon\, \top = \mathrm{id}\ \top : C_1\}$
$\{\epsilon\,(X \otimes Y) = \epsilon\, X \otimes \epsilon\, Y : C_1\ [X : C_0 \quad Y : C_0]\}$
$\leadsto$ Relations
$\{T\, G\, f = G\, f : C_1 \quad G\, T\, f = T\, f = C_1\ [f : C_1]\}$
$\{\eta\, T\, X = \mathrm{id}\ T\, X : C_1 \quad \epsilon\, G\, X = \mathrm{id}\ G\, X : C_1$
$T\,\eta\, X = \mathrm{id}\ T\, X : C_1 \quad T\,\epsilon\, X = (\eta\, X) \circ (\epsilon\, X) : C_1$
$G\,\eta\, X = (\eta\, X) \circ (\epsilon\, X) : C_1 \quad G\,\epsilon\, X = \mathrm{id}\ G\, X : C_1$
$[X : C_0]\}$

1.3.2 Extents

Given an SM 2-comprehension ($\mathbf{C}$, T, G, η, ϵ) (Section 1.3.1), we define, modifying [Pav90], an *extent* functor $\chi : \mathbf{C} \to 2 \multimap \mathbf{C}$ which commutes with the canonical $\mathbf{M}$ actions (and is thus a 2-natural transformation). Set $\chi = \eta \circ \epsilon$. By the definition of cotensor (Appendix 1.E), the natural transformation

$$\begin{tikzcd} \mathbf{C} \arrow[d, bend right=50, "G"'] \arrow[d, bend left=50, "T"] \arrow[d, phantom, "\overset{\chi}{\Longrightarrow}"] \\ \mathbf{C} \end{tikzcd}$$

defines a unique functor $\chi : \mathbf{C} \to 2 \multimap \mathbf{C}$ such that $\pi\ \chi = \chi$. This boils down to

$$\chi\,(f : X \to Y) \quad = \quad \begin{tikzcd} G\,X \arrow[r, "G\,f"] \arrow[d, "\chi\,X"'] & G\,Y \arrow[d, "\chi\,Y"] \\ T\,X \arrow[r, "T\,f"'] & T\,Y \end{tikzcd}$$

Proposition 1.3.2.1
The functor χ commutes with the canonical $\mathbf{M}$ *actions.*

Proof. E.g. $\chi\ T\ X = (\chi\ T\ X : G\ T\ X \to T\ T\ X) = (\mathrm{id}\, T\ X : T\ X \to T\,X) = T\,\chi\,X$. □

1.3.3 Dyadic Numbers

N (Section 1.2.3) is the disjoint union $\{0\} \cup \{2n+1 \mid n \in N\} \cup \{2n+2 \mid n \in N\}$. Thus each $n \in N$ is uniquely represented by a dyadic (= base 2 with digits 1, 2) numeral. The initial model in set of

$$N \qquad \{0 : N\} \qquad \{s_1\, n : N\ [n : N]\} \qquad \{s_2\, n : N\ [n : N]\}$$

is

$$\top \xrightarrow{0} N \xrightarrow{s_1} N \xrightarrow{s_2} N$$

where $s_1\, n = 2n + 1$, $s_2\, n = 2n + 2$. Thus the term $s_{k_0}\, s_{k_1}\, \ldots\, s_{k_{n-1}}\, 0$ can be identified with the dyadic numeral $x = k_{n-1}\, \ldots\, k_1\, k_0$ of *length* $|x| = n$.

1.3.4 Tiers

In $2 \multimap$ set we have the 2 tiers of dyadics

$$\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0 \quad = \quad \begin{tikzcd} \top \arrow[r, "0"] \arrow[d] & N \arrow[r, "s_k"] \arrow[d] & N \arrow[d] \\ \top \arrow[r] & \top \arrow[r] & \top \end{tikzcd}$$

$$\top \xrightarrow{0} N_1 \xrightarrow{s_k} N_1 \quad = \quad \begin{tikzcd} \top \arrow[r, "0"] \arrow[d] & N \arrow[r, "s_k"] \arrow[d, "\mathrm{id}"] & N \arrow[d, "\mathrm{id}"] \\ \top \arrow[r, "0"'] & N \arrow[r, "s_k"'] & N \end{tikzcd}$$

These satisfy

$$T\,(\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0) \quad = \quad \top \xrightarrow{\mathrm{id}} \top \xrightarrow{\mathrm{id}} \top$$

$$G\,(\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0) \quad = \quad \top \xrightarrow{0} N_1 \xrightarrow{s_k} N_1$$

1.4 Linear Time

1.4.1 A Linear Time Doctrine

The objects of the linear time doctrine $\mathfrak{LinTime}$ are (or see the specification below)

1. SM 2-comprehensions $(\mathbf{C}, T, G, \eta, \epsilon)$ (Section 1.3.1)

2. with dyadics
$$\top \xrightarrow{0} N_0 \xrightarrow{s_1} N_0 \xrightarrow{s_2} N_0$$
in $\mathbf{C}$ such that
$$T\,(\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0) \qquad = \qquad \top \xrightarrow{\mathrm{id}} \top \xrightarrow{\mathrm{id}} \top$$
and satisfying

3. Leivant's flat recursion (as below) and

4. Bloch's very safe recursion (as below).

We set
$$G\,(\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0) \qquad = \qquad \top \xrightarrow{0} N_1 \xrightarrow{s_k} N_1$$

Thus $T\,N_1 = T\,G\,N_0 = G\,N_0 = N_1$.

The tier 0 category $\mathbf{C}_{(0)}$, which we need for 3. and 4., has as objects the $\mathbf{C}$ commuting

$$\begin{tikzcd}
X & T\,X \arrow[d, shift left, "i"] \\
& \top \arrow[u, shift left, "i_1"]
\end{tikzcd}$$

and as maps $(X, i, i_1) \to (X', i', i_1')$ the $\mathbf{C}$ commuting

$$\begin{tikzcd}
X \arrow[r, "f"] & X' & T\,X \arrow[r, "T\,f"] \arrow[d, "i"'] & T\,X' \arrow[d, "i'"] \\
& & \top \arrow[r, "\mathrm{id}"'] & \top
\end{tikzcd}$$

$\mathbf{C}_{(0)}$ has SM structure by taking $\top$ to be

$$\begin{tikzcd}
\top & T & \top \arrow[dl, "\mathrm{id}"] \\
 & \top \arrow[u, "\mathrm{id}"] &
\end{tikzcd}$$

$(f : (X,\ i,\ i_1) \rightarrow (X',\ i',\ i'_1)) \otimes (Y,\ j,\ j_1)$ to be

$$\begin{tikzcd}
X \otimes Y \arrow[r, "f \otimes Y"] & X' \otimes Y & T\,X \otimes T\,Y \arrow[r, "T\,f \otimes T\,Y"] \arrow[d, "i \otimes j"'] & T\,X' \otimes T\,Y \arrow[d, "i' \otimes j"] \\
 & & \top \otimes \top \arrow[d, "\lambda\,\top"'] & \top \otimes \top \arrow[d, "\lambda\,\top"] \\
 & & \top \arrow[r, "\mathrm{id}"'] & \top
\end{tikzcd}$$

and e.g. $\sigma\ (X,\ i,\ i_1)\ (Y,\ j,\ j_1)$ to be

$$\begin{tikzcd}
X \otimes Y \arrow[r, "\sigma\,X\,Y"] & Y \otimes X & T\,X \otimes T\,Y \arrow[r, "\sigma\,T\,X\,T\,Y"] \arrow[d, "i \otimes j"'] & T\,Y \otimes T\,X \arrow[d, "j \otimes i"] \\
 & & \top \otimes \top \arrow[r, "\sigma\,\top\,\top = \mathrm{id}"] \arrow[d, "\lambda\,\top"'] & \top \otimes \top \arrow[d, "\lambda\,\top"] \\
 & & \top \arrow[r, "\mathrm{id}"'] & \top
\end{tikzcd}$$

Coherence for $\mathbf{C}_{(0)}$ follows from Mac Lane's coherence theorem [Tro92].

Flat recursion [Lei94] is that (using a remark of R. Cockett's) $\forall$ $\mathbf{C}_{(0)}$ objects $(X,\ i,\ i_1)$, the

$$\begin{tikzcd}
& N_0 \otimes X \arrow[d, "s_1 \otimes X"] & \\
\top \otimes X \arrow[r, "0 \otimes X"'] & N_0 \otimes X & N_0 \otimes X \arrow[l, "s_2 \otimes X"]
\end{tikzcd}$$

$$\begin{tikzcd}
& \top \otimes T\,X \arrow[d, "\mathrm{id}"] & \\
\top \otimes T\,X \arrow[r, "\mathrm{id}"'] & \top \otimes T\,X \arrow[d, "\top \otimes i"] & \top \otimes T\,X \arrow[l, "\mathrm{id}"] \\
& \top \otimes \top \arrow[d, "\lambda\,\top"] & \\
& \top &
\end{tikzcd}$$

are sum cocones in $\mathbf{C}_{(0)}$. (This is case analysis as in Section 1.3.3. The tail of chosen isomorphisms is part of being in $\mathbf{C}_{(0)}$.) In other words, *flat recursion* is that $\forall$ $\mathbf{C}$ commuting (with $k = 1,\ 2$)

$$\begin{tikzcd}
X \arrow[r, "g"] & Y
\end{tikzcd}
\qquad
\begin{tikzcd}
N_0 \otimes X \arrow[r, "h_k"] & Y
\end{tikzcd}$$

$$\begin{tikzcd}
T\,X \arrow[r, "T\,g"] \arrow[d, shift left, "i"] & T\,Y \arrow[d, shift left, "j"] \\
\top \arrow[r, "\mathrm{id}"'] \arrow[u, shift left, "i_1"] & \top \arrow[u, shift left, "j_1"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\top \otimes T\,X \arrow[r, "T\,h_k"] \arrow[d, "\top \otimes i"'] & T\,Y \arrow[d, "j"] \\
\top \otimes \top \arrow[r, "\lambda\,\top"'] & \top
\end{tikzcd}$$

$\exists!$ $\mathbf{C}$ commuting

$$\begin{tikzcd}
\top \otimes X \arrow[r, "0 \otimes X"] \arrow[d, "\lambda\,X"'] & N_0 \otimes X \arrow[r, "s_k \otimes X"] \arrow[d, "f"'] \arrow[dr, "h_k"'] & N_0 \otimes X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y & Y
\end{tikzcd}
\qquad
\begin{tikzcd}
\top \otimes T\,X \arrow[r, "T\,f"] \arrow[d, "\top \otimes i"'] & T\,Y \arrow[d, "j"] \\
\top \otimes \top \arrow[r, "\lambda\,\top"'] & \top
\end{tikzcd}$$

In particular we define *delete* $D : N_0 \to N_0$ such that (dropping o's), for $n : \top \to N_0$,

$$D\,0 = 0 \qquad D\,s_k\,n = n$$

by the flat recursion

$$\begin{tikzcd}
\top \arrow[r, "0"] \arrow[dr, "0"'] & N_0 \arrow[r, "s_k"] \arrow[d, "D"'] \arrow[dr, "\mathrm{id}"'] & N_0 \arrow[d, "D"] \\
 & N_0 & N_0
\end{tikzcd}$$

Very safe recursion [Blo92] is that $\forall$ C commuting (with $k = 1, 2$)

$$\begin{tikzcd}
X \arrow[r, "g"] & Y \arrow[r, "h_k"] & Y
\end{tikzcd}
\qquad
\begin{tikzcd}
T\,Y \arrow[r, "T\,h_k"] \arrow[d, shift left, "j"] & T\,Y \arrow[d, shift left, "j"] \\
\top \arrow[r, "\mathrm{id}"'] \arrow[u, shift left, "j_1"] & \top \arrow[u, shift left, "j_1"]
\end{tikzcd}$$

$\exists!$ C commuting

$$\begin{tikzcd}
\top \otimes X \arrow[r, "0 \otimes X"] \arrow[d, "\lambda\, X"'] & N_1 \otimes X \arrow[r, "s_k \otimes X"] \arrow[d, "f"] & N_1 \otimes X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y \arrow[r, "h_k"'] & Y
\end{tikzcd}$$

(Compare this with Proposition 1.2.4.2.)

In particular we define tier 1 diagonal $\delta : N_1 \to N_0 \otimes N_0$ and concatenation $\bullet : N_1 \otimes N_0 \to N_0$ such that (dropping o's and ignoring λ $\top$, ϵ N_0), for $n : \top \to N_1$, $x : \top \to N_0$,

$$\delta\, n = n \otimes n$$
$$0 \bullet x = x \qquad (s_k\, n) \bullet x = s_k\,(n \bullet x)$$

by the very safe recursions

$$\begin{tikzcd}
\top \arrow[r, "0"] & N_1 \arrow[r, "s_k"] \arrow[d, "\delta"] & N_1 \arrow[d, "\delta"] \\
\top \otimes \top \arrow[u, "\lambda\, \top"] \arrow[r, "0 \otimes 0"'] & N_0 \otimes N_0 \arrow[r, "s_k \otimes s_k"'] & N_0 \otimes N_0
\end{tikzcd}$$

$$\begin{tikzcd}
\top \otimes N_0 \arrow[r, "0 \otimes N_0"] \arrow[d, "\lambda\, N_0"'] & N_1 \otimes N_0 \arrow[r, "s_k \otimes N_0"] \arrow[d, "\bullet"] & N_1 \otimes N_0 \arrow[d, "\bullet"] \\
N_0 \arrow[r, "\mathrm{id}"'] & N_0 \arrow[r, "s_k"'] & N_0
\end{tikzcd}$$

Putting together the layers, the linear time doctrine $\mathfrak{LinTime}$, both objects and maps, is specified by Sections 1.1.3, 1.2.1, 1.2.2, 1.3.1, Appendix 1.C, and the following.

$\rightsquigarrow$ Dyadics

$$\begin{array}{l}
\{N_0 : C_0 \quad 0 : C_1 \quad d\,0 = \top : C_0 \quad c\,0 = N_0 : C_0 \\
\ s_1 : C_1 \quad d\,s_1 = N_0 : C_0 \quad c\,s_1 = N_0 : C_0 \\
\ s_2 : C_1 \quad d\,s_2 = N_0 : C_0 \quad c\,s_2 = N_0 : C_0 \\
\ T\,0 = \mathrm{id}\ \top : C_1 \quad T\,s_1 = \mathrm{id}\ \top : C_1 \quad T\,s_2 = \mathrm{id}\ \top : C_1 \\
\ N_1 : C_0 \quad N_1 = G\,N_0 : C_0\}
\end{array}$$

$\rightsquigarrow$ Flat recursion

$$\begin{array}{l}
\{!\ R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1 : C_1 \\
\quad d\,R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1 = N_0 \otimes d\,g : C_0 \\
\quad (R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1) \circ (0 \otimes d\,g) = g \circ \lambda\,d\,g : C_1 \\
\quad (R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1) \circ (s_1 \otimes d\,g) = h_1 : C_1 \\
\quad (R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1) \circ (s_2 \otimes d\,g) = h_2 : C_1 \\
\quad j \circ (T\,R_0\,g\,h_1\,h_2\,i\,i_1\,j\,j_1) = (\lambda\,\top) \circ (\top \otimes i) : C_1 \\
\quad [j \circ T\,g = i : C_1 \\
\quad\ j \circ T\,h_1 = (\lambda\,\top) \circ (\top \otimes i) : C_1 \quad j \circ T\,h_2 = (\lambda\,\top) \circ (\top \otimes i) : C_1 \\
\quad\ i_1 \circ i = \mathrm{id}\ T\,d\,g : C_1 \quad i \circ i_1 = \mathrm{id}\ \top : C_1 \\
\quad\ j_1 \circ j = \mathrm{id}\ T\,c\,g : C_1 \quad j \circ j_1 = \mathrm{id}\ \top : C_1 \\
\quad\ d\,h_1 = N_0 \otimes d\,g : C_0 \quad c\,h_1 = c\,g : C_0 \\
\quad\ d\,h_2 = N_0 \otimes d\,g : C_0 \quad c\,h_2 = c\,g : C_0 \\
\quad\ g : C_1 \quad h_1 : C_1 \quad h_2 : C_1 \quad i : C_1 \quad i_1 : C_1 \quad j : C_1 \quad j_1 : C_1]\}
\end{array}$$

$$\begin{array}{l}
\rightsquigarrow \text{Very safe recursion}\\
\{!\ R_1\ g\ h_1\ h_2\ j\ j_1 : C_1\\
\quad d\ R_1\ g\ h_1\ h_2\ j\ j_1 = N_1 \otimes d\ g : C_0\\
\quad (R_1\ g\ h_1\ h_2\ j\ j_1) \circ ((G\ 0) \otimes d\ g) = g \circ \lambda\ d\ g : C_1\\
\quad (R_1\ g\ h_1\ h_2\ j\ j_1) \circ ((G\ s_1) \otimes d\ g) = h_1 \circ R_1\ g\ h_1\ h_2\ j\ j_1 : C_1\\
\quad (R_1\ g\ h_1\ h_2\ j\ j_1) \circ ((G\ s_2) \otimes d\ g) = h_2 \circ R_1\ g\ h_1\ h_2\ j\ j_1 : C_1\\
\quad [j \circ T\ h_1 = j : C_1 \quad j \circ T\ h_2 = j : C_1\\
\quad\ j_1 \circ j = \mathrm{id}\ T\ c\ g : C_1 \quad j \circ j_1 = \mathrm{id}\ \top : C_1\\
\quad\ d\ h_1 = c\ g : C_0 \quad c\ h_1 = c\ g : C_0\\
\quad\ d\ h_2 = c\ g : C_0 \quad c\ h_2 = c\ g : C_0\\
\quad\ g : C_1 \quad h_1 : C_1 \quad h_2 : C_1 \quad j : C_1 \quad j_1 : C_1]\}
\end{array}$$

1.4.2 Formal Linear Time

By the arguments in Appendix 1.B and Section 1.4.1, there is an initial category **I** in 𝔏𝔦𝔫𝔗𝔦𝔪𝔢. Further, **I** is the quotient of a Herbrand universe $\mathrm{colim}_i\ P_i$. We call the **I** maps *formal linear time* maps and think of their representatives in the Herbrand universe as programs. The *standard model* $\Gamma_2 = (2 \multimap \Gamma) \circ \chi$ of these formal maps is the composition

$$\begin{tikzcd}
\mathbf{I} \arrow[rr, "\Gamma_2"] \arrow[dr, "\chi"'] & & 2 \multimap \mathsf{set} \\
& 2 \multimap \mathbf{I} \arrow[ur, "2 \multimap \Gamma"'] &
\end{tikzcd}$$

of the extent χ (Section 1.3.2) and the cotensor with 2 of Γ, where Γ is

$$\begin{array}{c}
\Gamma : \mathbf{I} \to \mathsf{set}\\
X \mapsto \mathbf{I}(\top,\ X) = \{\mathbf{I}\ \text{map}\ f \mid d\ f = \top,\ c\ f = X\}\\
f \mapsto f \circ \_
\end{array}$$

(Thus Γ consists of the no inputs formal maps.)

Proposition 1.4.2.1
For **I** *initial in the doctrine* 𝔏𝔦𝔫𝔗𝔦𝔪𝔢 *(Section 1.4.1)*

1. $\Gamma\ \top = \{\mathrm{id}\ \top\}$.

2. $\Gamma\,(X \otimes Y) = \{(x \otimes y) \circ (\lambda\, \top)^{-1} \mid x \in \Gamma\, X,\ y \in \Gamma\, Y\}$.

3. $\Gamma\, N_0 = \{\mathrm{std}_0\; n \mid n \in N\}$, *where* $\mathrm{std}_0\; 0 = 0$, $\mathrm{std}_0\,(2n + k) = s_k \circ \mathrm{std}_0\; n$.

4. *Up to natural isomorphism, the unique* $\mathfrak{LinTime}$ *functor* $i : \mathbf{I} \to 2 \multimap \mathbf{set}$ *is* Γ_2.

Proof. See Appendix 1.F. □

Proposition 1.4.2.2
The linear time functions $N^I \to N^{I'}$ *on deterministic multi-tape Turing machines with constant numbers of tapes are precisely those of the form* $\Gamma\ G\ f$ *for* $\mathbf{I}$ *maps* f, *where* $\mathbf{I}$ *is initial in the doctrine* $\mathfrak{LinTime}$ *(Section 1.4.1).*

Proof. See Sections 1.4.3–1.4.7. □

1.4.3 Dyadic Register Machines

A *dyadic register machine* has state consisting of

instruction pointer	i
data vector	d
program vector	p

and has the run algorithm

```
$i := 0$
load inputs into some of the $d_j$
zero the rest of the $d_j$
load the program into the $p_i$
while $i < |p|$
    execute $p_i$
look at outputs among the $d_j$
```

The vectors d, p have (big enough) finite lengths $|d|$, $|p|$ and their components d_j, p_i for $0 \le j < |d|$, $0 \le i < |p|$ are registers. The i, d_j contain natural numbers. The p_i contain instructions. The instructions and their actions are

$s_k\ j\ a$	$d_j := s_k\ d_j \quad i := a$
$D\ j\ a$	$d_j := D\ d_j \quad i := a$
$C\ j\ a\ b\ c$	$i := a$ if $d_j = 0$ $i := b$ if d_j is odd $i := c$ if d_j is non-zero even

where $s_1\ n = 2n+1$, $s_2\ n = 2n+2$, $D\ 0 = 0$, $D\ (2n+1) = n$, $D\ (2n+2) = n$. (Compare this with Section 1.3.3.)

1.4.4 Turing Machines

We show that the dyadic register machines (Section 1.4.3) and the deterministic multi-tape Turing machines with constant numbers of tapes [BDG88, BC94] compute the same linear time numeric functions.

1. Suppose we have an n tape deterministic Turing machine with tape alphabet $\{\#,\ 1,\ 2\}$, where $\#$ denotes blank. Simulate the tape

$$t_j = \ \ldots\ \#\ a_{-m}\ \ldots\ a_{-2}\ a_{-1}\ \underset{\triangle}{a_0}\ a_1\ \ldots\ a_{n-1}\ \#\ \ldots$$

where a_{-m} and a_{n-1} are the leftmost and rightmost non-blank symbols and $\triangle$ is the head, by a pair of data registers containing the dyadic numerals $d_{2j} = a'_{-m}\ \ldots\ a'_{-2}\ a'_{-1}$ and $d_{2j+1} = a'_{n-1}\ \ldots\ a'_1\ a'_0$, where a'_j codes a_j by

a_j	#	1	2
a'_j	11	12	22

E.g. simulate move head j right by popping a'_0 from d_{2j+1} (viewed as a stack) and then pushing a'_0 to d_{2j} and simulate halt by (out of range) address $|p|$.

2. We simulate a dyadic register machine $(i,\ d,\ p)$ by a $|d|$ tape deterministic Turing machine with tape alphabet $\{\#,\ 1,\ 2\}$. Simulate the data register $d_j = a_{n-1}\ \ldots\ a_1\ a_0$ by the tape

$$t_j = \ \ldots\ \#\ \underset{\triangle}{a_0}\ a_1\ \ldots\ a_{n-1}\ \#\ \ldots$$

E.g. simulate $s_k\ d_j\ a$ by

```
move head $j$ left
write $k$ at head $j$
if $a < |p|$
  set control state to $a$
else
  halt
```

3. The simulations 1., 2. change execution times only by constant factors, but the numbers change formats. However, reversal and alternate 2 (un)padding can be done in linear time on either of these machine models (with enough registers or tapes). Further, reformatting needs to be done only at input and output.

1.4.5 Enough Maps

We will show that every linear time numeric function has the form $\Gamma\ G\ f'$ for some **I** map f'. We do this by coding dyadic register machines (Section 1.4.3) inside **I** (Section 1.4.2). We drop ∘'s and work modulo α's and λ's [JS91, Jay89].

In fact, with inputs $X = N_1^{\otimes I}$ and state (instruction pointer and data) and program $Y = N_0^{\otimes(1+|d|+|p|)}$ (where e.g. ${N_1}^{\otimes 0} = \top$, ${N_1}^{\otimes(I+1)} = N_1 \otimes {N_1}^{\otimes I}$), we will code the initialization by an **I** map $g : X \to Y$, and the next state transition by an **I** map $h : Y \to Y$. The state vector $f\ n \otimes x_0 \otimes x_1\ \ldots$, for $n : \top \to N_1$, $x_i : \top \to N_1$, is then defined by the very safe recursion

$$\begin{tikzcd}
\top \otimes X \arrow[r, "0 \otimes X"] \arrow[d, "\lambda\, X"'] & N_1 \otimes X \arrow[r, "s_k \otimes X"] \arrow[d, "f"] & N_1 \otimes X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y \arrow[r, "h"'] & Y
\end{tikzcd}$$

So we will also code a big enough linear time bound by an **I** map $t : X \to N_1$. We then use σ (Section 1.2.2) and $G\ \delta$ (Section 1.4.1) to define $\delta : X \to X \otimes X$ such that, for $x : \top \to X$, $\delta\ x = x \otimes x$. Putting all this together, we define f'

by the composition

$$\begin{tikzcd} X \arrow[r, "\delta"] \arrow[rrr, bend right=20, "f'"'] & X \otimes X \arrow[r, "t \otimes X"] & N_1 \otimes X \arrow[r, "G\,f"] & G\,Y \end{tikzcd}$$

Actually, we still need to get the outputs out of Y. But tensor together an id N_0 for each output and an $\eta\ N_0$ for each non-output.

We define g as a tensor of $\epsilon\ N_0$'s, to load inputs, 0's, to zero work space, and $s_{k_0}\ s_{k_1}\ \ldots$ 0's, to load program instructions.

We use s_1 and $\bullet$ (really $G\ \bullet$ with $\bullet$ from Section 1.4.1) together with σ and $G\ \delta$ to define $t : X \rightarrow N_1$ such that, for $x_i : \top \rightarrow N_i$,

$$t\ x_0 \otimes x_1\ \cdots = s_1\ s_1\ \ldots\ x_0 \bullet x_0\ \ldots\ x_1 \bullet x_1\ \ldots$$

With enough s_1's and $\bullet$'s, t will output any given linear time in the sense that (using Section 1.3.3 and Proposition 1.4.2.1)

$$|t\ x_0 \otimes x_1\ \ldots| = \sum_i A_i|x_i| + B$$

for any given $A_i,\ B \in N$.

Finally, we define h by coding execute p_i. This last looks at a constant amount of low end (= least significant) digits and then, depending on what it sees, modifies a constant amount of low end digits. We code this as follows.

1. Use σ to permute a y_j (the state vector is $y_0 \otimes y_1 \otimes \cdots : \top \rightarrow Y$ by Proposition 1.4.2.1) into position (the 0th) for an R_0.

2. Use R_0 to destructively read:

   $$R_0\ \cdots : N_0 \otimes Y' \rightarrow Y = N_0 \otimes Y'$$

   In order to not destroy more than 1 digit of y_j, route $D\ y_j$ to the 0th position. (R_0's cases can 'see' this much of the 0th position.)

3. Undo the permutation of 1.

4. Use 1.–3. to destructively 'address decode'.

5. At the leaves of the 'address decode' of 3. place a 'rom' of tensored actions $s_{k_0}\ s_{k_1}\ \ldots\ D\ D\ \ldots$ to modify, and possibly restore, low end digits.

1.4.6 Safety

The essential feature of the Bellantoni-Cook composition [BC92] is *safety*: that tier 0 inputs ($\top \to N_0$) can not affect tier 1 outputs ($N_1 \to$). We now show safety in **I**. Consider **I** map $f : N_1^{\otimes I} \otimes N_0^{\otimes J} \to N_1^{\otimes I'}$. Applying $\eta : \mathrm{id} \to T$ we get (as $T\ N_0 = \top$ and $\eta\ \top = \mathrm{id}$) commuting

$$\begin{tikzcd}
N_1^{\otimes I} \otimes N_0^{\otimes J} \arrow[r, "f"] \arrow[d, "\eta -"'] & N_1^{\otimes I'} \arrow[d, "\mathrm{id}"] \\
N_1^{\otimes I} \otimes \top^{\otimes J} \arrow[r, "T\ f"'] & N_1^{\otimes I'}
\end{tikzcd}$$

1.4.7 Not Too Many Maps

We will compile **I** maps f (actually their Herbrand universe representatives) to dyadic register machine codes which compute $\Gamma\ G\ f$. At the same time we will bound the time and space used. We work modulo α's and λ's.

By Proposition 1.4.2.1, **I** maps $\top \to N_1^{\otimes I} \otimes N_0^{\otimes J}$ decompose as **I** maps $x_i : \top \to N_1$, $y_j : \top \to N_0$ and we can identify the components x_i, y_j with dyadic numerals. We call the components x_i, y_j variables and write $|x_i|$, $|y_j|$ for their lengths (Section 1.3.3). Thus given

$$f : N_1^{\otimes I} \otimes N_0^{\otimes J} \to N_1^{\otimes I'} \otimes N_0^{\otimes J'}$$

we have

$$x'_0 \otimes x'_1\ \ldots\ y'_0 \otimes y'_1\ \cdots = f \circ (x_0 \otimes x_1\ \ldots\ y_0 \otimes y_1\ \ldots)$$

and by safety (Section 1.4.6)

$$x'_0 \otimes x'_1\ \cdots = f \circ (x_0 \otimes x_1\ \ldots\ 0 \otimes 0\ \ldots)$$

We will show that

$$\mathrm{time}(f \circ (x_0 \otimes x_1\ \ldots\ y_0 \otimes y_1\ \ldots)) \leq \sum_i A_i |x_i| + B$$

where this is the computation time not counting zeroing. We will account for zeroing separately. (Alternatively, [Blo92] codes so that, at the expense of producing a linear amount of garbage, zeroing can be done in constant time.) To show this time bound, we will also need the output bounds, for $i' \in I'$, $j' \in J'$,

$$|x'_{i'}| \leq \textstyle\sum_i A_i|x_i| + B$$
$$|y'_{j'}| \leq \textstyle\sum_i A_i|x_i| + \max_j |y_j| + B$$

(We use max to get by with just 1 set of A_i, $B \in N$ for each f.)

By Appendix 1.B, **I** is built up by id, $\circ$, $\otimes$, α, σ, λ, $\top$, G, η, ϵ, R_0, R_1 from N_0, 0, s_1, s_2 (with the codomain and domain c, d reducing away). We do induction on this build up.

We wish to compute e.g. $y_0 \otimes y_1 \mapsto y_1 \otimes (0 \circ \eta\ N_0)$ in, except for the zeroing of y_0, constant time. Thus we represent variables y_j by pairs of data registers

$$\begin{array}{|c|}\hline 'y_j \\ \hline y_j \\ \hline\end{array}$$

where names $'y_j$ have constant length while values y_j need not. Then we implement $y_0 \otimes y_1 \mapsto y_1 \otimes (0 \circ \eta\ N_0)$ by

$$\begin{array}{|c|}\hline 'y_0 \\ \hline y_0 \\ \hline 'y_1 \\ \hline y_1 \\ \hline\end{array} \mapsto \begin{array}{|c|}\hline 'y_1 \\ \hline 0 \\ \hline 'y_0 \\ \hline y_1 \\ \hline\end{array}$$

We compile symmetries σ and variables x_i similarly.

We do not so compile $y_0 \otimes y_1 \mapsto y_1 \otimes y_1$, which is not in **I**. It may be impossible to implement $y_0 \otimes y_1 \mapsto y_1 \otimes y_1$ in constant time on deterministic multi-tape Turing machines with constant numbers of tapes. The *diagonal issue* consists of not accounting for this.

We implement α, λ, id, ϵ by doing nothing. We implement T, G by doing nothing different. In bounds involving G, y_j's may become x_i's, but not conversely. This works by weakening max (or nothing) to +.

We implement s_1, s_2 by the instructions s_1, s_2.

We implement $f_1 \circ f_0$, where f_0's output variables are f_1's input variables, by first running the compilation of f_0 and then that of f_1. Even though f_0

may grow the inputs to f_1, the induction step works due to safety and the inductive hypothesis for output bounds.

We implement $f_0 \otimes f_1$, where f_0's variables are disjoint from those of f_1, by running the compilation of one of them and then that of the other.

We implement $(R_0\ h\ h_1\ h_2\ \ldots) \circ (y_0 \otimes x_0 \otimes x_1\ \ldots\ y_1 \otimes y_2\ \ldots)$ using instructions C, D:

$$\begin{aligned} & C\ y_0\ g\ h_1'\ h_2' \\ h_1' :\ & D\ y_0\ h_1 \\ h_2' :\ & D\ y_0\ h_2 \end{aligned}$$

We implement $(R_1\ g\ h_1\ h_2\ \ldots) \circ (x_0 \otimes x_1\ \ldots\ y_0 \otimes y_1\ \ldots)$ using loops. Since $(R_1\ g\ h_1\ h_2\ \ldots) \circ ((s_{k_0} \circ s_{k_1} \circ \ldots 0) \otimes \ldots) = h_{k_0} \circ h_{k_1} \circ \ldots g \circ \ldots$ we first need to reverse control digits:

$$\begin{array}{|c|}\hline 'x_0 \\ \hline \ldots\ k_1\ k_0 \\ \hline \end{array} \mapsto \begin{array}{|c|}\hline '\mathsf{tmp} \\ \hline k_0\ k_1\ \ldots \\ \hline \end{array}$$

So, with the scratch data register tmp initially zeroed, the code is

$$\begin{aligned} f :\ & C\ x_0\ g\ \mathsf{rev}_1\ \mathsf{rev}_2 \\ \mathsf{rev}_1 :\ & s_1\ \mathsf{tmp}\ f' \\ \mathsf{rev}_2 :\ & s_2\ \mathsf{tmp}\ f' \\ f' :\ & D\ x_0\ f \\ g :\ & \ldots \\ & \ldots\quad \mathsf{lp} \\ \mathsf{lp} :\ & C\ \mathsf{tmp}\ \mathsf{next}\ h_1\ h_2 \\ h_1 :\ & \ldots \\ & \ldots\quad \mathsf{lp}' \\ h_2 :\ & \ldots \\ & \ldots\quad \mathsf{lp}' \\ \mathsf{lp}' :\ & D\ \mathsf{tmp}\ \mathsf{lp} \end{aligned}$$

Being tier 0, h_1, h_2 take (ignoring zeroing) constant time. Thus

$$\mathrm{time}((R_1\ g\ h_1\ h_2\ \ldots) \circ (x_0 \otimes x_1\ \ldots)) \leq K''|x_0| + \sum_{i>0} A_i^g |x_i| + K'''$$

$$|y'_{j'}| \leq K|x_0| + \sum_{i>0} A_i^g |x_i| + \max_j |y_j| + K'$$

for constants K, K', K'', $K''' \in N$. (There are no $x'_{i'}$'s.)

Finally, we consider zeroing

$$\begin{tikzcd} N_0 \arrow[r, "\eta N_0"] & \top \arrow[r, "0"] & N_0 \end{tikzcd}$$

The code

$$\begin{aligned} f &: \; C \; y \; \mathsf{next} \; f' \; f' \\ f' &: \; D \; y \; f \end{aligned}$$

takes time $2|y| + 1$. Inside a loop R_1 only the first zeroing of y needs to be charged this much time. Additional zeroing just counteracts writings s_k within the loop. Thus we can divide up the time of this additional zeroing and charge it, in advance and in constant amounts, to the writings s_k within the loop.

1.5 The Linear Time Hierarchy

As a corollary of the proof (Sections 1.4.3–1.4.7) of Proposition 1.4.2.2, we will characterize the linear time hierarchy relations. As usual, we simulate constant numbers of alternations by adding data registers to deterministic machines. However, by typing these added registers tier 0 we implicitly, rather than explicitly, provide the needed bounds.

Proposition 1.5.1
The linear time hierarchy relations are precisely those of the form

$$\{(a_0, a_1, \ldots) \in N^I \mid \exists\, y_0 \,\forall\, y_1 \,\ldots\, \Gamma\, G\,(f \circ (x_0 \otimes x_1 \,\ldots\, y_0 \otimes y_1 \,\ldots)) = 0\}$$

for **I** *maps* $f : N_1^{\otimes I} \otimes N_0^{\otimes J} \to N_0$, $x_i = \mathrm{std}_1\, a_i : \top \to N_1$, $y_j : \top \to N_0$, *where* **I** *is initial in the doctrine* $\mathfrak{LinTime}$ *(Section 1.4.1).*

Proof. 1. One gets enough relations as one does for the polynomial time hierarchy [BC94, BDG88, BDG90]. Pop digits from y_j to decide the 'or' or 'and' branching. Increment j at alternations between 'or' and 'and'.

2. Only the loops R_1 can pop off and read more than a constant number of digits from the y_j. But, by the arguments in Section 1.4.7, the number of these iterations is linear in the $|x_i|$. Thus the quantifications $Q\ y_j$ are implicitly linearly bounded by the $|x_i|$. Thus we do not get too many relations. □

1.A Sketches as Presheaves

Suppose that V' is a model of ZFC [Jec78, Kun80]. With **Set** the category of sets and functions in V' and o denoting opposite, we have the power set functor $P : \mathbf{Set}^o \to \mathbf{Set}$. With the singleton maps $s_X : X \to P\ X$ by $x \mapsto \{x\}$, we have a cumulative hierarchy U : ordinals $\to V'$ by

$$U_0 = \{\} \qquad s_{U_i} : U_i \to U_{i+1} = P\ U_i \qquad U_j = \operatorname*{colim}_{i<j} U_i$$

We assume there exist strongly inaccessible cardinals $\alpha_0 < \alpha_1 < \ldots$. Then the U_{α_i} are models of ZFC. (Thus the U_{α_i} are Grothendieck universes [Bor94]. With $\eta : \mathrm{id} \to P^2$ the unit of the adjunction $P^o \dashv P$, can the $\eta_{U_i} : U_i \to U_{i+1} = P^2\ U_i$, which are natural, be used in place of the $s_{U_i} : U_i \to U_{i+1} = P\ U_i$, which are not?) We say that the sets in U_{α_i} are α_i *small* and we write $\mathbf{set}_i$ for the category of sets and functions in U_{α_i}. **set** (small) will be whichever of $\mathbf{set}_0$, $\mathbf{set}_1$, ... (α_0 small, α_1 small, ...) is convenient.

Given a category **S** and an object s of $\mathbf{set}^{\mathbf{S}}$, we say that s is *finite* iff the disjoint union $\sum_{\mathbf{S}\text{ object }X} s\ X$ is finite. An object X of **S** has *finite fan-out* iff $\mathbf{S}(X, \_)$ is finite. A *sketch theory* is a small category **S** such that

1. All **S** objects have finite fan-out.

2. **S** is *acyclic* (= 1-way). I.e. all endomorphisms in **S** are identities.

3. **S** is *skeletal*. I.e. isomorphic objects in **S** are equal.

(2. and 3. are due to F. Lawvere. In particular, finite partial orders such as **2**, V, and **3** are sketch theories.)

Proposition 1.A.1
Given a sketch theory S, $\exists$ *a function* height : $\mathrm{S}_{obj} \to N$, *from the objects of* S *to the natural numbers, such that* $\forall$ *non-identity* S *maps* $f : X \to X'$ *we have* height X' < height X.

Proof. Suppose that, $\forall$ S objects X, $\exists$ a largest $n \in N$ such that $\exists$ non-identity S maps

$$X \xrightarrow{f_0} \xrightarrow{f_1} \cdots \xrightarrow{f_{n-1}}$$

Set height $X = n$. Then

$$X \xrightarrow{f} X' \xrightarrow{f_0} \xrightarrow{f_1} \cdots \xrightarrow{f_{n'-1}}$$

shows that height X' < height X.

Suppose that $\exists$ an S object X and an ω indexed chain

$$X \xrightarrow{f_0} \xrightarrow{f_1} \cdots$$

of non-identity S maps. Then, with $\prod_{i<0} f_i = \mathrm{id}_X$, $\prod_{i<j+1} f_i = f_j \circ \prod_{i<j} f_i$, as X has finite fan-out, $\exists\ j < k$ such that $\prod_{i<j} f_i = \prod_{i<k} f_i$. Thus $\prod_{i<k-j} f_{j+i}$ is an endomorphism. Therefore f_i is an isomorphism, and thus an identity, contrary to hypothesis. □

So, taking S objects as sorts and enough S maps as operators, sketch theories S are the equational specifications of Section 1.1.1. Thus S sketches and homomorphisms form the category $\mathrm{set}^{\mathrm{S}}$, which is the category of presheaves on S°.

Proposition 1.A.2
For a sketch theory S, *the finite* S *sketches are precisely the finitely presentable objects [AR94] in* $\mathrm{set}^{\mathrm{S}}$.

Proof. Finite colimits of representables in $\mathrm{set}^{\mathrm{S}}$ are finite. □

1.B Initial Models

Given a basic almost equational specification (S, M) (Section 1.1.2), we will construct (in set) an initial model I of (S, M). Then we will show, using projective covers, that I is a quotient of a Herbrand universe [Llo87]. (Elsewhere we will show that the set models of the (S, M) are precisely the locally finitely presentable categories.)

Suppose that for each $m \in M$ we add m^* to M, where

$$\begin{tikzcd}
{} \arrow[r, "m"] \arrow[d, "m"'] \arrow[dr, phantom, "po"] & {} \arrow[d] \arrow[ddr, "id"] & \\
{} \arrow[r] \arrow[drr, "id"'] & {} \arrow[dr, "m^*"] & \\
& & {}
\end{tikzcd}$$

commutes and po denotes push-out. Then, to check for orthogonality, it is enough to check for injectivity [AR94]. (A sketch s is injective relative to M iff maps to s extend along $m \in M$. The m^* then force the uniqueness.)

So we add the m^* to M. For $m \in M$, the *instances* $\widetilde{m}$ of m are from the push-outs

$$\begin{tikzcd}
{} \arrow[r, "\widetilde{m}"] \arrow[dr, phantom, "po"] & {} \\
{} \arrow[u] \arrow[r, "m"'] & {} \arrow[u]
\end{tikzcd}$$

With 0 the sketch empty at each S sort and the $\widetilde{m}_i$ instances of elements of M, I is the colimit of the tree of all finite deductions from 0 [Mak94, AR94], a fragment of which is

$$\begin{tikzcd}
0 \arrow[r, "\widetilde{m}_0"] \arrow[dr, "\widetilde{m}_2"'] & s_0 \arrow[r, "\widetilde{m}_1"] \arrow[dr, "\widetilde{m}_3"'] & s_1 \\
& s_2 & s_3
\end{tikzcd}$$

Call sums of representable sketches (=, up to isomorphism, the S(X, _)) *free*. As the representables are projective, relative to the regular epimorphisms, so are the frees. (An object P is *projective* relative to a set of maps E iff maps from P lift over $e \in E$, where $\tilde{h}$ lifts h over e iff

$$\begin{tikzcd}
P \arrow[d, "h"'] \arrow[dr, "\tilde{h}"] & \\
{} & {} \arrow[l, "e"]
\end{tikzcd}$$

commutes.)

Given a sketch s, call $x \in s\ X$, with X an S sort, *primitive* iff it is not of the form $(s\ f)\ x'$ for some S operator $f\ x' : X\ [x' : X']$ and some $x' \in s\ X'$. With P the sum of representables $s(X,\ \_)$ as indexed by the primitive elements $x \in s\ X$ (i.e. $[\ldots\ x : X\ \ldots]$ for primitive $x \in s\ X$), a finite sketch s has the projective cover $c : P \to s$, where c assigns parameter x to primitive element x. (A *projective cover* $c : P \to s$ is a regular epimorphism c such that P is projective, relative to the regular epimorphisms, and such that $\forall$ commuting

$$\begin{tikzcd} {} \arrow[r, "i"] \arrow[dr, "e"'] & P \arrow[d, "c"] \\ & s \end{tikzcd}$$

if e is a regular epimorphism and i is a monomorphism, then i is an isomorphism.)

From a projective cover $c : P \to s$ we can form a kernel pair

$$\begin{tikzcd} \tilde{s} \arrow[r, "k"] \arrow[d, "h"'] \arrow[dr, phantom, "\text{pb}", pos=0.2] & P \arrow[d, "c"] \\ P \arrow[r, "c"'] & s \end{tikzcd}$$

where pb denotes pull-back. Then, when s is finite, we can take a projective cover $\tilde{c} : \tilde{P} \to \tilde{s}$ of $\tilde{s}$. Thus homomorphisms between finite sketches lift as

$$\begin{tikzcd} \tilde{P} \arrow[r, "\tilde{c}"] \arrow[d] & \tilde{s} \arrow[r, shift left, "h"] \arrow[r, shift right, "k"'] \arrow[d] & P \arrow[r, "c"] \arrow[d] & s \arrow[d, "\tilde{m}"] \\ \tilde{P}' \arrow[r, "\tilde{c}'"'] & \tilde{s}' \arrow[r, shift left, "h'"] \arrow[r, shift right, "k'"'] & P' \arrow[r, "c'"'] & s' \end{tikzcd}$$

With this presentation

$$\begin{tikzcd} \tilde{P}_i \arrow[r, shift left, "h_i \circ \tilde{c}_i"] \arrow[r, shift right, "k_i \circ \tilde{c}_i"'] & P_i \arrow[r, "c_i"] & s_i \end{tikzcd}$$

of the tree of finite deductions from 0, I is a quotient of the Herbrand universe $\operatorname{colim}_i P_i$.

1.C Coherence

The *pentagon*, *triangle*, and *hexagon* conditions are that, with ⊗'s and the arguments of the α, σ, λ omitted,

$$\begin{tikzcd}
W\,(X\,(Y\,Z)) \arrow[r, "\alpha"] \arrow[d, "W\,\alpha"'] & (W\,X)\,(Y\,Z) \arrow[r, "\alpha"] & ((W\,X)\,Y)\,Z \arrow[d, "\alpha\,Z"] \\
W\,((X\,Y)\,Z) \arrow[rr, "\alpha"'] & & (W\,(X\,Y))\,Z
\end{tikzcd}$$

$$\begin{tikzcd}
\top\,(X\,Y) \arrow[rr, "\alpha"] \arrow[dr, "\lambda"'] & & (\top\,X)\,Y \arrow[dl, "\lambda\,Y"] \\
& X\,Y &
\end{tikzcd}$$

$$\begin{tikzcd}
X\,(Y\,Z) \arrow[r, "X\,\sigma"] \arrow[d, "\alpha"'] & X\,(Z\,Y) \arrow[r, "\alpha"] & (X\,Z)\,Y \arrow[d, "\sigma\,Y"] \\
(X\,Y)\,Z \arrow[r, "\sigma"'] & Z\,(X\,Y) \arrow[r, "\alpha"'] & (Z\,X)\,Y
\end{tikzcd}$$

commute. We specify these by

$$\begin{aligned}
&\{(\alpha\,W\,(X\otimes Y)\,Z)\circ(W\otimes(\alpha\,X\,Y\,Z))\\
&\ = ((\alpha\,W\,X\,Y)\otimes Z)\circ(\alpha\,(W\otimes X)\,Y\,Z)\circ(\alpha\,W\,X\,(Y\otimes Z)) : C_1\\
&\ [W:C_0 \quad X:C_0 \quad Y:C_0 \quad Z:C_0]\}\\
&\{\lambda\,(X\otimes Y) = ((\lambda\,X)\otimes Y)\circ(\alpha\,\top\,X\,Y) : C_1\,[X:C_0 \quad Y:C_0]\}\\
&\{(\alpha\,Z\,X\,Y)\circ(\sigma\,(X\otimes Y)\,Z)\circ(\alpha\,X\,Y\,Z)\\
&\ = ((\sigma\,X\,Z)\otimes Y)\circ(\alpha\,X\,Z\,Y)\circ(X\otimes(\sigma\,Y\,Z)) : C_1\\
&\ [X:C_0 \quad Y:C_0 \quad Z:C_0]\}
\end{aligned}$$

1.D Linear Implication

Small vector spaces over a field k (or small modules over a commutative ring k) form not only an SM category (with unit $\top = k$), but an SMC category, i.e. ∀ objects X ∃ a *linear implication* adjunction

$$\_\otimes X \dashv X \multimap \_$$

For vector spaces over k, $X \multimap Y = \hom(X, Y)$, where $\hom(X, Y)$ is the set of k-linear maps from X to Y given the point-wise vector space structure. Indeed, for vector spaces one defines the tensor $\otimes$ by

$$\_ \otimes X \dashv \hom(X, \_)$$

Having all the adjunctions

$$\begin{array}{ccc}
\mathbf{C}(\_ \otimes X, Y) & \approx & \mathbf{C}(\_, X \multimap Y) \\
@ \circ (g \otimes X) & & g \\
f & \mapsto & \Lambda f = (X \multimap f) \circ \kappa
\end{array}$$

is specified by

$$\begin{array}{l}
\{X \multimap Y : C_0 \quad @\, X\, Y : C_1 \\
\;\; d\, @\, X\, Y = (X \multimap Y) \otimes X : C_0 \quad c\, @\, X\, Y = Y : C_0 \\
\;\; [X : C_0 \quad Y : C_0]\} \\
\{!\, \Lambda\, W\, X\, f : C_1 \\
\quad d\, \Lambda\, W\, X\, f = W : C_0 \quad c\, \Lambda\, W\, X\, f = X \multimap (c\, f) : C_0 \\
\quad (@\, X\, (c\, f)) \circ ((\Lambda\, W\, X\, f) \otimes X) = f : C_1 \\
\quad [d\, f = W \otimes X : C_0 \quad W : C_0 \quad X : C_0 \quad f : C_1]\} \\
\{\kappa\, W\, X : C_1 \quad \kappa\, W\, X = \Lambda\, W\, X\, \mathrm{id}\,(W \otimes X) : C_1 \\
\;\; [W : C_0 \quad X : C_0]\} \\
\{X \multimap h : C_1 \\
\;\; X \multimap h = \Lambda\, (X \multimap (d\, h))\, X\, (h \circ @\, X\, (d\, h)) : C_1 \\
\;\; [X : C_0 \quad h : C_1]\}
\end{array}$$

1.E Cotensor

The 2-category *cat* of small categories is the prototypical example of a 2-category [MP89, Bor94], with 0-cells = small categories, 1-cells = functors, and 2-cells = natural transformations. Whereas a category $\mathbf{C}$ has hom sets $\mathbf{C}(X, Y) = \{\mathbf{C}\text{ map } f \mid d\ f = X,\ c\ f = Y\}$, a 2-category $\mathfrak{D}$ has hom categories $\mathfrak{D}(X, Y)$. Vertical composition is inside hom categories, while horizontal composition is of hom categories. Thus horizontal composition is similar to the tensor ⊙ of SM categories. Indeed, both 2-categories and SM categories are special cases of bicategories.

Implicit in the linear time doctrine $\mathfrak{LinTime}$ (Section 1.4.1) is the 2-category $\mathfrak{SM}$ of small SM categories with witnessed structure. The models (in set) of the specification of Sections 1.1.3, 1.2.1, 1.2.2 and Appendix 1.C give $\mathfrak{SM}$ except for the 2-cells. As 2-cells we take those natural transformations ν preserving $\top$, $\otimes$ up to identity in the sense that

$$\nu\,\top = \mathrm{id} \qquad \nu\,(X \otimes Y) = \nu\,X \otimes \nu\,Y$$

In a 2-category $\mathfrak{D}$ with 0-cell $\mathbf{C}$, the cotensor $2 \multimap \mathbf{C}$ is defined by a 2-natural isomorphism [Bor94, Kel89]

$$\mathfrak{D}(\_, \mathbf{C})^2 \qquad \approx \qquad \mathfrak{D}(\_, 2 \multimap \mathbf{C})$$

$$\begin{tikzcd}
- \arrow[d, bend right=50] \arrow[d, bend left=50] \arrow[d, phantom, "\overset{\nu}{\Longrightarrow}"] & & - \arrow[d, bend right=50] \arrow[d, bend left=50] \arrow[d, phantom, "\overset{\nu}{\Longrightarrow}"] \\
2 \multimap \mathbf{C} \arrow[d, bend right=50] \arrow[d, bend left=50] \arrow[d, phantom, "\overset{\pi}{\Longrightarrow}"] & \mapsto & 2 \multimap \mathbf{C} \\
\mathbf{C} & &
\end{tikzcd}$$

In $\mathfrak{cat}$, $2 \multimap \mathbf{C}$ exists and is the functor category $\mathbf{C}^2$ with the additional structure of the comma category $\mathbf{C}/\mathbf{C}$. There π views an object of $\mathbf{C}^2$ as a map of $\mathbf{C}$.

Proposition 1.E.1
$2 \multimap \mathbf{C}$ exists in $\mathfrak{SM}$ and has the underlying category $\mathbf{C}^2$.

Proof. Form $2 \multimap \mathbf{C}$ in $\mathfrak{cat}$. As $\top$ take $\mathrm{id} : \top \to \top$. As $\otimes$ on

$$\begin{tikzcd}
X_0 \arrow[r, "f_0"] \arrow[d, "x"'] & X_0' \arrow[d, "x'"] & Y_0 \arrow[d, "y"] \\
X_1 \arrow[r, "f_1"'] & X_1' & Y_1
\end{tikzcd}$$

take

$$\begin{tikzcd}
X_0 \otimes Y_0 \arrow[r, "f_0 \otimes Y_0"] \arrow[d, "X \otimes Y"'] & X'_0 \otimes Y_0 \arrow[d, "X' \otimes Y"] \\
X_1 \otimes Y_1 \arrow[r, "f_1 \otimes Y_1"'] & X'_1 \otimes Y_1
\end{tikzcd}$$

As e.g. $\sigma\ X\ Y$ take

$$\begin{tikzcd}
X_0 \otimes Y_0 \arrow[r, "\sigma\ X_0\ Y_0"] \arrow[d, "X \otimes Y"'] & Y_0 \otimes X_0 \arrow[d, "Y \otimes X"] \\
X_1 \otimes Y_1 \arrow[r, "\sigma\ X_1\ Y_1"'] & Y_1 \otimes X_1
\end{tikzcd}$$

□

1.F Gluing

Gluing (= Freyd covers) [LS86] is a partial alternative to reduction techniques. First we apply gluing to I initial in 𝔏𝔦𝔫𝔗𝔦𝔪𝔢 and then to I initial in 𝔗.

Proof of Proposition 1.4.2.1. Form in 𝔠𝔞𝔱 the comma category

$$\begin{tikzcd}
(2 \multimap \mathbf{set})/\Gamma_2 \arrow[r, "\pi_1"] \arrow[d, "\pi_0"', ""{name=A}] & \mathbf{I} \arrow[d, "\Gamma_2", ""'{name=B}] \\
2 \multimap \mathbf{set} \arrow[r, "\mathrm{id}"'] & 2 \multimap \mathbf{set}
\arrow[Rightarrow, from=A, to=B]
\end{tikzcd}$$

We proceed to add structure to $(2 \multimap \mathbf{set})/\Gamma_2$ in such a way that π_1 will end up in 𝔏𝔦𝔫𝔗𝔦𝔪𝔢.

Maps $f : X \to Y$ in $(2 \multimap \mathbf{set})/\Gamma_2$ are $2 \multimap \mathbf{set}$ commuting

$$\begin{tikzcd}
\widetilde{X} \arrow[r, "\widetilde{f}"] \arrow[d, "\widetilde{x}"'] & \widetilde{Y} \arrow[d, "\widetilde{y}"] \\
\Gamma_2\, X \arrow[r, "\Gamma_2\, f"'] & \Gamma_2\, Y
\end{tikzcd}$$

In set this is commuting

$$\begin{tikzcd}
\widetilde{X}_0 \arrow[rr] \arrow[dr] \arrow[dd] & & \tilde{Y}_0 \arrow[dr] \arrow[dd, dashed] & \\
& \widetilde{X}_1 \arrow[rr] \arrow[dd] & & \tilde{Y}_1 \arrow[dd] \\
\Gamma\, G\, X \arrow[rr, dashed] \arrow[dr] & & \Gamma\, G\, Y \arrow[dr, dashed] & \\
& \Gamma\, T\, X \arrow[rr] & & \Gamma\, T\, Y
\end{tikzcd}$$

with faces

cover (= π_0 projection)

$$\begin{tikzcd}
\widetilde{X}_0 \arrow[r, "\tilde{f}_0"] \arrow[d, "\tilde{x}"'] & \tilde{Y}_0 \arrow[d, "\tilde{Y}"] \\
\widetilde{X}_1 \arrow[r, "\tilde{f}_1"'] & \tilde{Y}_1
\end{tikzcd}$$

base

$$\begin{tikzcd}
\Gamma\, G\, X \arrow[r, "\Gamma\, G\, f"] \arrow[d, "\Gamma\, \chi\, X"'] & \Gamma\, G\, Y \arrow[d, "\Gamma\, \chi\, Y"] \\
\Gamma\, T\, X \arrow[r, "\Gamma\, T\, f"'] & \Gamma\, T\, Y
\end{tikzcd}$$

> which 'remembers' the π_1 projection to the **I** map $f : X \to Y$. (We have overloaded the symbol f to denote both a ($2 \multimap$ set)/Γ_2 map and an **I** map).

stage 0

$$\begin{tikzcd}
\widetilde{X}_0 \arrow[r, "\tilde{f}_0"] \arrow[d, "\tilde{x}_0"'] & \tilde{Y}_0 \arrow[d, "\tilde{y}_0"] \\
\Gamma\, G\, X \arrow[r, "\Gamma\, G\, f"'] & \Gamma\, G\, Y
\end{tikzcd}$$

stage 1

$$\begin{tikzcd}
\widetilde{X}_1 \arrow[r, "\tilde{f}_1"] \arrow[d, "\tilde{x}_1"'] & \tilde{Y}_1 \arrow[d, "\tilde{y}_1"] \\
\Gamma\, T\, X \arrow[r, "\Gamma\, T\, f"'] & \Gamma\, T\, Y
\end{tikzcd}$$

(and 2 more).

As $\top$ take

$$\begin{tikzcd}
\top \arrow[r] \arrow[d, "0\mapsto\mathrm{id}"'] & \top \arrow[d, "0\mapsto\mathrm{id}"] \\
\Gamma\,\top \arrow[r, "\mathrm{id}"'] & \Gamma\,\top
\end{tikzcd}$$

(Recall that $\chi\ \top = \mathrm{id}$ and that in set $\top = \{0\}$.)

As $(f : X \to X') \otimes Y$ take, writing $\tilde{x} \otimes \tilde{y}$ for what is really $(x,\ y) \mapsto (\tilde{x}\,x \otimes \tilde{y}\,y) \circ (\lambda\ \top)^{-1}$,

cover

$$\begin{tikzcd}
\widetilde{X}_0 \times \widetilde{Y}_0 \arrow[r, "\tilde{f}_0\times\widetilde{Y}_0"] \arrow[d, "\widetilde{X}\times\widetilde{Y}"'] & \widetilde{X}'_0 \times \widetilde{Y}_0 \arrow[d, "\widetilde{X}'\times\widetilde{Y}"] \\
\widetilde{X}_1 \times \widetilde{Y}_1 \arrow[r, "\tilde{f}_1\times\widetilde{Y}_1"'] & \widetilde{X}'_1 \times \widetilde{Y}_1
\end{tikzcd}$$

base

$$\begin{tikzcd}
\Gamma\,G\,(X \otimes Y) \arrow[r, "\Gamma\,G\,(f\otimes Y)"] \arrow[d, "\Gamma\,\chi\,(X\otimes Y)"'] & \Gamma\,G\,(X' \otimes Y) \arrow[d, "\Gamma\,\chi\,(X'\otimes Y)"] \\
\Gamma\,T\,(X \otimes Y) \arrow[r, "\Gamma\,T\,(f\otimes Y)"'] & \Gamma\,T\,(X' \otimes Y)
\end{tikzcd}$$

stage 0

$$\begin{tikzcd}
\widetilde{X}_0 \times \widetilde{Y}_0 \arrow[r] \arrow[d, "\tilde{x}_0\otimes\tilde{y}_0"'] & \widetilde{X}'_0 \times \widetilde{Y}_0 \arrow[d, "\tilde{x}'_0\otimes\tilde{y}_0"] \\
\Gamma\,G\,(X \otimes Y) \arrow[r] & \Gamma\,G\,(X' \otimes Y)
\end{tikzcd}$$

stage 1

$$\begin{tikzcd}
\widetilde{X}_1 \times \widetilde{Y}_1 \arrow[r] \arrow[d, "\tilde{x}_1\otimes\tilde{y}_1"'] & \widetilde{X}'_1 \times \widetilde{Y}_1 \arrow[d, "\tilde{x}'_1\otimes\tilde{y}_1"] \\
\Gamma\,T\,(X \otimes Y) \arrow[r] & \Gamma\,T\,(X' \otimes Y)
\end{tikzcd}$$

E.g. as σ take

cover

$$\begin{tikzcd}
\widetilde{X}_0 \times \widetilde{Y}_0 \arrow[r, "{(x,y)\mapsto(y,x)}"] \arrow[d] & \widetilde{Y}_0 \times \widetilde{X}_0 \arrow[d] \\
\widetilde{X}_1 \times \widetilde{Y}_1 \arrow[r, "{(x,y)\mapsto(y,x)}"'] & \widetilde{Y}_1 \times \widetilde{X}_1
\end{tikzcd}$$

base

$$\begin{tikzcd}
\Gamma\, G\,(X \otimes Y) \arrow[r, "\Gamma\, G\, \sigma\, X\, Y"] \arrow[d, "\Gamma\, \chi\, (X\otimes Y)"'] & \Gamma\, G\,(Y \otimes X) \arrow[d, "\Gamma\, \chi\, (Y\otimes X)"] \\
\Gamma\, T\,(X \otimes Y) \arrow[r, "\Gamma\, T\, \sigma\, X\, Y"'] & \Gamma\, T\,(Y \otimes X)
\end{tikzcd}$$

For ϵ, η take

cover

$$\begin{tikzcd}
\widetilde{X}_0 \arrow[r, "\mathrm{id}"] \arrow[d, "\mathrm{id}"'] & \widetilde{X}_0 \arrow[r, "\widetilde{X}"] \arrow[d, "\widetilde{X}"] & \widetilde{X}_1 \arrow[d, "\mathrm{id}"] \\
\widetilde{X}_0 \arrow[r, "\widetilde{X}"'] & \widetilde{X}_1 \arrow[r, "\mathrm{id}"'] & \widetilde{X}_1
\end{tikzcd}$$

base

$$\begin{tikzcd}
\Gamma\, G\, X \arrow[r, "\mathrm{id}"] \arrow[d, "\mathrm{id}"'] & \Gamma\, G\, X \arrow[r, "\Gamma\, \chi\, X"] \arrow[d, "\Gamma\, \chi\, X"] & \Gamma\, T\, X \arrow[d, "\mathrm{id}"] \\
\Gamma\, G\, X \arrow[r, "\Gamma\, \chi\, X"'] & \Gamma\, T\, X \arrow[r, "\mathrm{id}"'] & \Gamma\, T\, X
\end{tikzcd}$$

For N_0 take

$$\begin{tikzcd}
N \arrow[r] \arrow[d, "\mathrm{std}_1"'] & \mathrm{T} \arrow[d, "0\mapsto\mathrm{id}"] \\
\Gamma\, N_1 \arrow[r, "\Gamma\, \chi\, N_0"'] & \Gamma\, \mathrm{T}
\end{tikzcd}$$

where $\mathrm{std}_1\ 0 = 0$, $\mathrm{std}_1\,(2n+1) = s_1 \circ \mathrm{std}_1\ n$, $\mathrm{std}_1\,(2n+2) = s_2 \circ \mathrm{std}_1\ n$.

For

$$\top \xrightarrow{0} N_0 \xrightarrow{s_k} N_0$$

take
stage 0

$$\begin{tikzcd}
\top \arrow[r, "0"] \arrow[d, "0\mapsto\mathrm{id}"'] & N \arrow[r, "s_k"] \arrow[d, "\mathrm{std}_1"] & N \arrow[d, "\mathrm{std}_1"] \\
\Gamma\, \top \arrow[r, "\Gamma\, 0"'] & \Gamma\, N_1 \arrow[r, "\Gamma\, s_k"'] & \Gamma\, N_1
\end{tikzcd}$$

stage 1

$$\begin{tikzcd}
\top \arrow[r] \arrow[d, "0\mapsto\mathrm{id}"'] & \top \arrow[r] \arrow[d] & \top \arrow[d] \\
\Gamma\, \top \arrow[r, "\mathrm{id}"'] & \Gamma\, \top \arrow[r, "\mathrm{id}"'] & \Gamma\, \top
\end{tikzcd}$$

Given $g : X \to Y$, $h_k : N_0 \otimes X \to Y$,

$$\begin{tikzcd}
T\, X \arrow[r, "T\, g"] \arrow[d, shift left, "i"] & T\, Y \arrow[d, shift left, "j"] \\
\top \arrow[r, "\mathrm{id}"'] \arrow[u, shift left, "i_1"] & \top \arrow[u, shift left, "j_1"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\top \otimes T\, X \arrow[r, "T\, h_k"] \arrow[d, "\top\otimes i"'] & T\, Y \arrow[d, "j"] \\
\top \otimes \top \arrow[r, "\lambda\, \top"'] & \top
\end{tikzcd}$$

for R_0 take
stage 0

$$\begin{tikzcd}
N \times \widetilde{X}_0 \arrow[r, "R_0\, \widetilde{g}_0\, \widetilde{h}_{10}\, \widetilde{h}_{20}"] \arrow[d, "\mathrm{std}_1 \otimes \widetilde{x}_0"'] & \widetilde{Y}_0 \arrow[d, "\widetilde{y}_0"] \\
\Gamma\, (N_1 \otimes G\, X) \arrow[r, "\Gamma\, G\, R_0\, g\, h_1\, h_2\, i\, i_1\, j\, j_1"'] & \Gamma\, G\, Y
\end{tikzcd}$$

stage 1

$$\begin{tikzcd}
\top \times \widetilde{X}_1 \arrow[r, "\widetilde{h}_{11}"] \arrow[d, "(0\mapsto\mathrm{id})\otimes\widetilde{x}_1"'] & \widetilde{Y}_1 \arrow[d, "\widetilde{y}_1"] \\
\Gamma\, (\top \otimes T\, X) \arrow[r, "\Gamma\, T\, R_0\, g\, h_1\, h_2\, i\, i_1\, j\, j_1"'] & \Gamma\, T\, Y
\end{tikzcd}$$

Given $g : X \to Y$, $h_k : Y \to Y$,

$$\begin{tikzcd}
T\,Y \arrow[r, "T\,h_k"] \arrow[d, shift left, "j"] & T\,Y \arrow[d, shift left, "j"] \\
\top \arrow[r, "\mathrm{id}"'] \arrow[u, shift left, "j_1"] & \top \arrow[u, shift left, "j_1"]
\end{tikzcd}$$

for R_1 take

stage 0

$$\begin{tikzcd}
N \times \widetilde{X_0} \arrow[r, "R_1\,\widetilde{g}_0\,\widetilde{h}_{10}\,\widetilde{h}_{20}"] \arrow[d, "\mathrm{std}_1 \otimes \widetilde{X}_0"'] & \widetilde{Y}_0 \arrow[d, dashed, "\widetilde{y}_0"] \\
\Gamma\,(N_1 \otimes G\,X) \arrow[r, "\Gamma\,G\,R_1\,g\,h_1\,h_2\,j\,j_1"'] & \Gamma\,G\,Y
\end{tikzcd}$$

stage 1

$$\begin{tikzcd}
N \times \widetilde{X_1} \arrow[r, "R_1\,\widetilde{g}_1\,\widetilde{h}_{11}\,\widetilde{h}_{21}"] \arrow[d, "\mathrm{std}_1 \otimes \widetilde{X}_1"'] & \widetilde{Y}_1 \arrow[d, dashed, "\widetilde{y}_1"] \\
\Gamma\,(N_1 \otimes T\,X) \arrow[r, "\Gamma\,T\,R_1\,g\,h_1\,h_2\,j\,j_1"'] & \Gamma\,T\,Y
\end{tikzcd}$$

Thus, since **I** is initial in $\mathfrak{LinTime}$, ∃! commuting

$$\begin{tikzcd}
& (2 \multimap \mathsf{set})/\Gamma_2 \arrow[dr, "\pi_1"] & \\
\mathbf{I} \arrow[ur, "j"] \arrow[rr, "\mathrm{id}"'] & & \mathbf{I}
\end{tikzcd}$$

in $\mathfrak{LinTime}$.

1. Start with $f : \top \to \top$ in **I**. Then, as $\pi_1 \circ j = \mathrm{id}$, $j\,f$ has stage 0

$$\begin{tikzcd}
\top \arrow[r] \arrow[d, "0 \mapsto \mathrm{id}"'] & \top \arrow[d, "0 \mapsto \mathrm{id}"] \\
\Gamma\,\top \arrow[r, "\Gamma\,G\,f"'] & \Gamma\,\top
\end{tikzcd}$$

So, starting from 0 in the the upper left, $G\,f = G\,f \circ \mathrm{id} = \mathrm{id}$, while

$$\begin{tikzcd}
\top \arrow[r, "G\,f"] \arrow[d, "\epsilon\,\top = \mathrm{id}"'] & \top \arrow[d, "\epsilon\,\top = \mathrm{id}"] \\
\top \arrow[r, "f"'] & \top
\end{tikzcd}$$

also commutes.

2. Start with $f : \top \to X \otimes Y$ in **I**. $j\,f$ has stage 0

$$\begin{tikzcd}
\top \arrow[r, "0 \mapsto (x,\, y)"] \arrow[d, "0 \mapsto \mathrm{id}"'] & \widetilde{X_0} \times \widetilde{Y_0} \arrow[d, "\tilde{x}_0 \otimes \tilde{y}_0"] \\
\Gamma\,\top \arrow[r, "\Gamma\,G\,f"'] & \Gamma\,(G\,X \otimes G\,Y)
\end{tikzcd}$$

Thus $G\,f = (\tilde{x}_0\,x \otimes \tilde{y}_0\,y) \circ (\lambda\,\top)^{-1}$, while

$$\begin{tikzcd}
\top \arrow[r, "G\,f"] \arrow[d, "\epsilon\,\top = \mathrm{id}"'] & G\,X \otimes G\,Y \arrow[d, "\epsilon\,X \otimes \epsilon\,Y"] \\
\top \arrow[r, "f"'] & X \otimes Y
\end{tikzcd}$$

also commutes.

3. Start with $f : \top \to N_0$ in **I**. $j\,f$ has stage 0

$$\begin{tikzcd}
\top \arrow[r, "0 \mapsto n"] \arrow[d, "0 \mapsto \mathrm{id}"'] & N \arrow[d, "\mathrm{std}_1"] \\
\Gamma\,\top \arrow[r, "\Gamma\,G\,f"'] & \Gamma\,N_1
\end{tikzcd}$$

Thus $G\,f = \mathrm{std}_1\,n$, while

$$\begin{tikzcd}
\top \arrow[r, "G\,f"] \arrow[d, "\epsilon\,\top = \mathrm{id}"'] & N_1 \arrow[d, "\epsilon\,N_0"] \\
\top \arrow[r, "f"'] & N_0
\end{tikzcd}$$

also commutes.

4. Inductively define a natural isomorphism $\nu : i \to \Gamma_2$ with $\nu\ \top$ by 1., $\nu\,(X \otimes Y)$ by 2., and $\nu\ N_i$ by 3. □

Proof of 1.-3. of Proposition 1.2.4.1. We modify the above 1.-3. argument. Use the comma category

$$\begin{tikzcd}
\mathsf{set}/\Gamma \arrow[r, "\pi_1"] \arrow[d, "\pi_0"'] \arrow[dr, phantom, "\Longrightarrow"] & \mathbf{I} \arrow[d, "\Gamma"] \\
\mathsf{set} \arrow[r, "\mathsf{id}"'] & \mathsf{set}
\end{tikzcd}$$

For $(\tilde{x} : \widetilde{X} \to \Gamma\, X) \multimap (\tilde{y} : \tilde{Y} \to \Gamma\, Y)$ use the natural bijection

$$\begin{tikzcd}
\widetilde{W} \times \widetilde{X} \arrow[r, "\tilde{f}"] \arrow[d, "\tilde{w} \otimes \tilde{x}"'] & \tilde{Y} \arrow[d, "\tilde{y}"] \\
\Gamma\,(W \otimes X) \arrow[r, "\Gamma\, f"'] & \Gamma\, Y
\end{tikzcd}$$

$$\begin{tikzcd}
\widetilde{W} \arrow[rr, shift left=2, "w \mapsto (x \mapsto \tilde{f}\ w\ x)"] \arrow[r] \arrow[d, "\tilde{w}"'] & {} \arrow[r] \arrow[d, "\mathsf{pb}"] & \tilde{Y}^{\widetilde{X}} \arrow[d, "\tilde{y}^{\widetilde{X}}"] \\
\Gamma\, W \arrow[r, "\Gamma\, \Lambda\, f"'] & \Gamma\,(X \multimap Y) \arrow[r, "\gamma"'] & (\Gamma\, Y)^{\widetilde{X}}
\end{tikzcd}$$

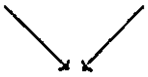
where $\gamma\ g = (x \mapsto @ \circ (g \otimes \tilde{x}\ x) \circ (\lambda\ \top)^{-1})$ and pb denotes pull-back. □

Chapter 2

V-Comprehensions and P Space

Introduction

Román [Rom89] characterized the primitive recursive functions as the image in set (the category of small sets) of categories initial in the doctrine which adds stable NNO to the FP doctrine (the doctrine of categories having finite products and witnessed structure). (Doctrines are often categories or 2-categories of categories.) We have characterized the linear time functions (Chapter 1) and the linear space, P time, and Kalmar elementary functions (Chapter 4) as the images in set^2 or set^3 of categories initial in doctrines. Here we characterize the P space functions as the image in set^V of categories initial in a doctrine, where V is the partial order



This tiered characterization of P space pumps up linear space by running machines longer. Instead, the tiered characterization of P space in [LM95] pumps up P time by making machines more powerful.

From set^2 and set^V we abstract (2- and V-) comprehensions (Section 2.1). Unary and dyadic NNO have the fragments flat and very safe recursions for SM comprehensions (Chapter 1) and flat and safe recursions for FP comprehensions (Section 2.2). (In Chapter 3 and elsewhere we will study working over

LCC and fibrational doctrines rather than over the SM and FP doctrines.) Our P space doctrine consists of FP V-comprehensions having unary flat recursion and compatible unary and dyadic safe recursions. In this doctrine V joins at tier 1 unary and dyadic numbers which have separate tier 0's. We show that the image in $\mathbf{set}^V$ is big enough (Section 2.3) by coding machines and running them long enough. We show that the image in $\mathbf{set}^V$ is small enough (Section 2.4) by inductively, relative to the Herbrand structure of the initial category (Chapter 1), deriving bounds on space, time, and output sizes. We also characterize (in Appendices 2.A, 2.B) the linear space and the P time functions using FP 2-comprehensions. Our P space doctrine glues these two doctrines together along the two sides of V.

We stand on many shoulders. For comprehensions we stand on [Pav90, JMS91, Law70], for initial categories and gluing (or Freyd covers) on [Rom89, LS86], for flat recursions on [Lei94], for very safe recursions on [Blo92], for safe recursions on [BC92, Bel92], for linear space on [Bel92, Rit63], for P time on [BC92, Cob65], and for P space on [Tho72, Huw76].

We write τ for terminal maps, and f, g for the tuple map of maps f and g having a common domain.

2.1 V-Comprehensions

2.1.1 Unary and Dyadic Numbers

Unary numbers are specified by sort and operators

$$N \qquad \{0 : N\} \qquad \{s\,x : N\,[x : N]\}$$

The initial model in set has

$$N = \{0,\ 1,\ 2,\ \ldots\} \qquad 0 = 0 \qquad s\,x = x + 1$$

Dyadic numbers are specified by sort and operators

$$N \qquad \{0 : N\} \qquad \{s_1\,x : N\,[x : N]\} \qquad \{s_2\,x : N\,[x : N]\}$$

The initial model in set has

$$N = \{0,\ 1,\ 2,\ \dots\} \qquad 0 = 0 \qquad s_k\,x = 2x + k \quad \text{for } k = 1,\ 2$$

2.1.2 2 Tier 0's

set^2 has the 2 tiers of numbers

$$\begin{tikzcd} N_0 \arrow[r, phantom, "="] & N \arrow[d] & N_1 \arrow[r, phantom, "="] & N \arrow[d, "\mathrm{id}"] \\ & 1 & & N \end{tikzcd}$$

with N_0 a quotient rather than a subobject of N_1. Linear space can be characterized using 2 tiers of unary numbers while P time can be characterized using 2 tiers of dyadic numbers (Appendices 2.A, 2.B). We will use some of the P time characterization to pump up linear space to P space by having 2 tier 0's, one unary and one dyadic. Thus our target will be set^V, rather than set^2, where V is the partial order $0.1 \;\rightarrow\; 1 \;\leftarrow\; 0.2$. In set^V we have the tiers of numbers

$$\begin{tikzcd} N_{0.1} \arrow[r, phantom, "="] & N \arrow[r] & 1 & 1 \arrow[l] \\ N_1 \arrow[r, phantom, "="] & N \arrow[r, "\mathrm{id}"] & N & N \arrow[l, "\mathrm{id}"'] \\ N_{0.2} \arrow[r, phantom, "="] & 1 \arrow[r] & 1 & N \arrow[l] \end{tikzcd}$$

2.1.3 Cotensor

The 2-category cat has 0-cells = (small) categories, 1-cells = functors, and 2-cells = natural transformations. The sub-2-category $\mathfrak{FP}$ of FP categories with witnessed structure has 0- and 1-cells as specified in a basic almost equational specification (Section 1.1) (in particular the functors are strict) and is full on 2-cells.

In cat the cotensors [Bor94, Kel89] $2 \multimap \mathbf{C}$ and $V \multimap \mathbf{C}$ exist and are $\mathbf{C}^2$ (with the additional structure of the comma category $\mathbf{C}/\mathbf{C}$) and $\mathbf{C}^V$. In $\mathfrak{FP}$ the cotensors $2 \multimap \mathbf{C}$ and $V \multimap \mathbf{C}$ again both exist and are $\mathbf{C}^2$ and $\mathbf{C}^V$.

2.1.4 2-Comprehensions

Since the ordinal 2 is the partial order $0 \to 1$, the endomorphisms end(2) of 2 form a partially ordered monoid. Reversing the multiplication, but not the partial order, set $\mathbf{M} = \text{end}(2)^{\circ}$. Abstracting from $2 \multimap \text{set}$ to $2 \multimap \mathbf{C}$ and then from the left $\mathbf{M}$ action on $2 \multimap \mathbf{C}$ induced by the right $\mathbf{M}$ action on 2, an *FP 2-comprehension* is a 2-functor $\mathbf{M} \to \mathfrak{FP}$. (For more details on 2-comprehensions see Section 1.3.)

2.1.5 V-Comprehensions

$\mathbf{M}_V$ is the sub partially ordered monoid of $\text{end}(V)^{\circ}$ generated by

$$\begin{tikzcd}
T_1 & = & 0.1 \arrow[dr] & 0.1 & G_1 & = & 0.1 \arrow[r] & 0.1 \\
& & 1 \arrow[r] & 1 & & & 1 \arrow[ur] & 1 \\
& & 0.2 \arrow[r] & 0.2 & & & 0.2 \arrow[uur] & 0.2 \\
T_2 & = & 0.1 \arrow[r] & 0.1 & G_2 & = & 0.1 \arrow[ddr] & 0.1 \\
& & 1 \arrow[r] & 1 & & & 1 \arrow[dr] & 1 \\
& & 0.2 \arrow[ur] & 0.2 & & & 0.2 \arrow[r] & 0.2
\end{tikzcd}$$

and full on 2-cells. An *FP V-Comprehension* is a 2-functor $\mathbf{M}_V \to \mathfrak{FP}$.

Proposition 2.1.5.1
As a monoid, $\mathbf{M}_V$ has generators T_1, G_1, T_2, G_2 and relations (for $i = 1, 2$)

$$\begin{aligned}
T_i^2 &= T_i \\
G_1\,T_2 = T_2\,G_1 = T_1\,G_1 &= G_2\,G_1 = G_1 \\
G_2\,T_1 = T_1\,G_2 = T_2\,G_2 &= G_1\,G_2 = G_2 \\
G_1\,T_1 = G_2\,T_2 &= T_2\,T_1 = T_1\,T_2
\end{aligned}$$

Proof. E.g. $G_1{}^2 = G_1\,(T_1\,G_1) = (G_1\,T_1)\,G_1 = (T_1\,T_2)\,G_1 = T_1\,(T_2\,G_1) = T_1\,G_1 = G_1$. □

Applying G_1, G_2, id, T_2, T_1, $T_1\,T_2$ to (0.1, 1, 0.2) we see that the partial order underlying $\mathbf{M}_V$ is (generated by)

$$\begin{tikzcd}
G_1 \arrow[dd, "\gamma_1"'] & & G_2 \arrow[dd, "\gamma_2"] \\
& \mathrm{id} \arrow[dl, "\eta_2"'] \arrow[dr, "\eta_1"] & \\
T_2 \arrow[dr, "\eta_1\,T_2"'] & & T_1 \arrow[dl, "\eta_2\,T_1"] \\
& T_1\,T_2 &
\end{tikzcd}$$

The compositions

$$\chi_1 = (\eta_1\,T_2) \circ \gamma_1 \qquad \chi_2 = (\eta_2\,T_1) \circ \gamma_2$$

will be important (Section 2.1.6).

Proposition 2.1.5.2
As a 2-category, in addition to the generators and relations of Proposition 2.1.5.1, $\mathbf{M}_V$ has generators η_1, γ_1, η_2, γ_2 and relations (for $i = 1, 2$ and with χ_1, χ_2 as just above)

$$\begin{array}{cc}
\eta_i\,T_i = T_i\,\eta_i = \mathrm{id} & \gamma_i\,T_i = \mathrm{id} \\
\gamma_1\,T_2 = T_2\,\gamma_1 = \gamma_1 & \gamma_2\,T_1 = T_1\,\gamma_2 = \gamma_2 \\
T_i\,\gamma_i = G_i\,\eta_i = \chi_i & G_i\,\gamma_i = \mathrm{id} \\
T_1\,\eta_2 = \eta_2\,T_1 & T_2\,\eta_1 = \eta_1\,T_2 \\
G_1\,\gamma_2 = \chi_2 & G_2\,\gamma_1 = \chi_1
\end{array}$$

Proof. From

$$\begin{tikzcd}
{} & {} \arrow[d, Rightarrow, "\eta_1"] & {} \arrow[ll, "\mathrm{id}"'] & {} & {} \arrow[d, Rightarrow, "\eta_2"] & {} \arrow[ll, "\mathrm{id}"'] \\
{} & {} & {} \arrow[ll, "T_1"] & {} & {} & {} \arrow[ll, "T_2"]
\end{tikzcd}$$

we have that

$$\begin{tikzcd}
\mathrm{id} \arrow[r, dashed, "\eta_2"] \arrow[d, "\eta_1"'] & T_2 \arrow[d, "\eta_1 T_2"] \\
T_1 \arrow[r, "T_1 \eta_2"'] & T_1 T_2
\end{tikzcd}$$

commutes and thus that $(\eta_2 T_1) \circ \eta_1 = (\eta_1 T_2) \circ \eta_2$.

E.g. $\eta_2 G_1 = \eta_2 T_2 G_1 = \mathrm{id}$, $G_2 \eta_1 = G_2 T_1 \eta_1 = \mathrm{id}$. □

Thus we can view an FP V-comprehension as strict FP functors

$$T_1,\ G_1,\ T_2,\ G_2 : \mathbf{C} \to \mathbf{C}$$

and natural transformations

$$\begin{tikzcd}
G_1 \arrow[dd, "\eta_1"'] & & G_2 \arrow[dd, "\eta_2"] \\
& \mathrm{id} \arrow[dl, "\eta_2"'] \arrow[dr, "\eta_1"] & \\
T_2 & & T_1
\end{tikzcd}$$

satisfying the relations of Propositions 2.1.5.1, 2.1.5.2.

With $\mathbf{M}_V^{\ddagger}$ the sub-2-category of $\mathbf{M}_V$ generated by T_1, T_2, η_1, η_2, the doctrine of 2-functors and 2-natural transformations

$$(\mathbf{C},\ T_1,\ T_2,\ \eta_1,\ \eta_2) : \mathbf{M}_V^{\ddagger} \to \mathfrak{FP}$$

allows generically adding $\mathbf{C}$ maps $1 \to X$, given $\mathbf{C}$ object X. For $\mathbf{C}$ in $\mathfrak{FP}$ one generically adds $1 \to X$ by passing, by pull-back along $X \to 1$, to the full subcategory $\mathbf{C}//X$ in $\mathbf{C}/X$ of the $\pi_0 : X \times Y \to X$ [LS86]. (One chooses FP structure on $\mathbf{C}//X$ such that the pull-back along $X \to 1$ is strict FP.) One defines T_i by $T_i\,(\pi_0 : X \times Y \to X) = (\pi_0 : X \times T_i\,Y \to X)$ and η_i by

$$\begin{tikzcd}
X \times Y \arrow[r] \arrow[d, "\pi_0"'] \arrow[rr, bend left, "(\eta_i X)\times(\eta_i Y)"] & X \times T_i\,Y \arrow[r] \arrow[d, "\pi_0"', "\mathrm{pb}"] & T_i\,X \times T_i\,Y \arrow[d, "\pi_0"] \\
X \arrow[r, "\mathrm{id}"'] & X \arrow[r, "\eta_i X"'] & T_i\,X
\end{tikzcd}$$

where pb denotes pull-back. This is implicit in the definition of safe recursion (Section 2.2).

2.1.6 Extents

Given an FP V-comprehension $(\mathbf{C}, T_k, G_k, \eta_k, \gamma_k)$, the $\mathbf{M}_V$ action on V also induces an FP V-comprehension $(V \multimap \mathbf{C}, T_k, G_k, \eta_k, \gamma_k)$. Using the definition of cotensor, we have a unique strict FP functor, which we call the *extent*,

$$\chi : \mathbf{C} \to V \multimap \mathbf{C}$$

such that 2-naturally

$$\mathfrak{FP}(\mathbf{C}, \mathbf{C})^V \approx \mathfrak{FP}(\mathbf{C}, V \multimap \mathbf{C})$$

$$\begin{tikzcd}[column sep=large]
\mathbf{C} \arrow[d, bend right=80, "G_1"', ""{name=A}] \arrow[d, "T_1 T_2" description, ""{name=B}] \arrow[d, bend left=80, "G_2", ""'{name=D}] \\
\mathbf{C}
\arrow[Rightarrow, from=A, to=B, "\chi_1"] \arrow[Rightarrow, from=D, to=B, "\chi_2"']
\end{tikzcd}
\quad \mapsto \quad
\begin{tikzcd}
\mathbf{C} \arrow[d, "\chi"] \\
V \multimap \mathbf{C}
\end{tikzcd}$$

Thus

$$\chi(f : X \to Y) \quad = \quad \begin{tikzcd}
G_1\,X \arrow[r, "G_1\,f"] \arrow[d, "\chi_1\,X"'] & G_1\,Y \arrow[d, "\chi_1\,Y"] \\
T_1\,T_2\,X \arrow[r, "T_1\,T_2\,f"] & T_1\,T_2\,Y \\
G_2\,X \arrow[r, "G_2\,f"'] \arrow[u, "\chi_2\,X"] & G_2\,Y \arrow[u, "\chi_2\,Y"']
\end{tikzcd}$$

Proposition 2.1.6.1
The extent functor χ is a 2-natural transformation between V-comprehensions.

Proof. We have the multiplication table

	T_1	G_1	T_2	G_2
G_1	$T_1\,T_2$	G_1	G_1	G_2
G_2	G_2	G_1	$T_1\,T_2$	G_2
$T_1\,T_2$	$T_1\,T_2$	G_1	$T_1\,T_2$	G_2

Thus e.g. χ applied to

$$\begin{tikzcd}
G_1\,X \arrow[r, "\eta_1\,X"] & T_2 X & X \arrow[l, "\eta_2\,X"']
\end{tikzcd}$$

is

$$\begin{tikzcd}
G_1\,X \arrow[r] \arrow[d] & G_1\,X \arrow[d] & G_1\,X \arrow[l] \arrow[d] \\
G_1\,X \arrow[r] & T_1\,T_2\,X & T_1\,T_2\,X \arrow[l] \\
G_1\,X \arrow[r] \arrow[u] & T_1\,T_2\,X \arrow[u] & G_2\,X \arrow[l] \arrow[u]
\end{tikzcd}$$

which is

$$\begin{tikzcd}
G_1\,\chi\,X \arrow[r] & T_2\,\chi\,X & \chi\,X \arrow[l]
\end{tikzcd}$$

with the arrows (including their names) working out right as the 2-cells in $\mathbf{M}_V$ are unique. □

2.1.7 Tiers

We abstract properties from $V \multimap$ set. There we have tier 0.1 unary

$$\begin{tikzcd}
1 \arrow[r, "0"] & N_{0.1} \arrow[r, "s"] & N_{0.1}
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
1 \arrow[r, "0"] \arrow[d] & N \arrow[r, "s"] \arrow[d] & N \arrow[d] \\
1 \arrow[r] & 1 \arrow[r] & 1 \\
1 \arrow[r] \arrow[u] & 1 \arrow[r] \arrow[u] & 1 \arrow[u]
\end{tikzcd}$$

tier 0.2 dyadic (for $k = 1, 2$)

$$\begin{tikzcd}
1 \arrow[r, "0'"] & N_{0.2} \arrow[r, "s_k"] & N_{0.2}
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
1 \arrow[r] \arrow[d] & 1 \arrow[r] \arrow[d] & 1 \arrow[d] \\
1 \arrow[r] & 1 \arrow[r] & 1 \\
1 \arrow[r, "0"'] \arrow[u] & N \arrow[r, "s_k"'] \arrow[u] & N \arrow[u]
\end{tikzcd}$$

and tier 1

$$N \quad = \quad \begin{tikzcd} N \arrow[d, "\mathrm{id}"] \\ N \\ N \arrow[u, "\mathrm{id}"'] \end{tikzcd}$$

Thus we have

$$\begin{array}{lll} T_1\ N_{0.1} = 1 & G_1\ N_{0.1} = N_1 & T_2\ N_{0.1} = N_{0.1} \\ T_1\ N_{0.2} = N_{0.2} & T_2\ N_{0.2} = 1 & G_2\ N_{0.2} = N_1 \end{array}$$

2.2 P Space

The objects of the P space doctrine $\mathfrak{PSpace}$ are (compare this with Section 1.4.1.)

1. FP V-comprehensions
$$(\mathbf{C},\ T_1,\ G_1,\ T_2,\ G_2,\ \eta_1,\ \gamma_1,\ \eta_2,\ \gamma_2)$$
2. with unary
$$1 \xrightarrow{0} N_{0.1} \xrightarrow{s} N_{0.1}$$
in $\mathbf{C}$ such that
$$T_1\ N_{0.1} = 1 \qquad T_2\ N_{0.1} = N_{0.1}$$
3. and dyadic (for $k = 1,\ 2$)
$$1 \xrightarrow{0'} N_{0.2} \xrightarrow{s_k} N_{0.2}$$
in $\mathbf{C}$ such that
$$T_1\ N_{0.2} = N_{0.2} \qquad T_2\ N_{0.2} = 1 \qquad G_2\ N_{0.2} = G_1\ N_{0.1}$$
satisfying
4. unary flat recursion (as below)

5. unary safe recursion (as below)

6. dyadic safe recursion (as below)

7. 5. and 6. are compatible (as below).

We set

$$\begin{aligned} G_1\,(1 \xrightarrow{0} N_{0.1} \xrightarrow{s} N_{0.1}) &\quad = \quad 1 \xrightarrow{0} N_1 \xrightarrow{s} N_1 \\ G_2\,(1 \xrightarrow{0'} N_{0.2} \xrightarrow{s_k} N_{0.2}) &\quad = \quad 1 \xrightarrow{0'} N_1 \xrightarrow{s_k} N_1 \end{aligned}$$

The tier $0.k$ categories $\mathbf{C}_{(0.k)}$ (for $k = 1,\ 2$), which we need for 4.–6., have as objects $\mathbf{C}$ commuting

$$\begin{tikzcd}
X & T_k\,X \arrow[d, shift left] \\
& 1 \arrow[u, shift left, "i"]
\end{tikzcd}$$

and as maps $(X,\ i) \to (X',\ i')$ $\mathbf{C}$ maps $f : X \to X'$. $\mathbf{C}_{(0.k)}$ has FP structure

$$\begin{tikzcd}
1 & = & 1 & T_k\,1 \arrow[d, shift left, dashed] \\
& & & 1 \arrow[u, shift left, "\mathrm{id}"]
\end{tikzcd}$$

$$\begin{tikzcd}
(X,\ i) \times (Y,\ j) & = & X \times Y & T_k\,X \times T_k\,Y \arrow[d, shift left] \\
& & & 1 \arrow[u, shift left, "{i,\,j}"]
\end{tikzcd}$$

Unary flat recursion [Lei94] is that (using a remark of R. Cockett's) $\forall$ $\mathbf{C}_{(0.1)}$ objects $(X,\ i)$

$$\begin{tikzcd}
& N_{0.1} \times X \arrow[d, "s \times X"] \\
1 \times X \arrow[r, "0 \times X"'] & N_{0.1} \times X
\end{tikzcd}$$

is a sum cocone in $\mathbf{C}_{(0.1)}$, i.e. $\forall$ C commuting

$$\begin{tikzcd}
X \arrow[r, "g"] & Y & N_{0.1} \times X \arrow[r, "h"] & Y
\end{tikzcd}$$

$$\begin{tikzcd}
T_1\, X \arrow[d, shift left] & T_1\, Y \arrow[d, shift left] \\
1 \arrow[u, shift left, "i"] & 1 \arrow[u, shift left, "j"]
\end{tikzcd}$$

$\exists!$ C commuting

$$\begin{tikzcd}
1 \times X \arrow[r, "0\times X"] \arrow[d, "\pi_1"'] & N_{0.1} \times X \arrow[r, "s\times X"] \arrow[d, "f"'] \arrow[dr, "h"'] & N_{0.1} \times X \arrow[d, "f"] \\
X \arrow[r, "g"'] & Y & Y
\end{tikzcd}$$

We write $R_{0.1}\ g\ h\ i\ j$ for this f.

Unary safe recursion [Bel92] is that $\forall$ C commuting

$$\begin{tikzcd}
X \arrow[r, "g"] & Y & X \times Y \arrow[r, "h"] & Y & X \times T_1\, Y \arrow[d, shift left, "\pi_0"] \\
 & & & & X \arrow[u, shift left, "{\mathrm{id}, j}"]
\end{tikzcd}$$

$\exists!$ C commuting

$$\begin{tikzcd}
1 \times X \arrow[r, "0\times X"] \arrow[d, "\pi_1"'] & N_1 \times X \arrow[r, "s\times X"] \arrow[d, "{\pi_1, f}"] & N_1 \times X \arrow[d, "{\pi_1, f}"] \\
X \arrow[r, "{\mathrm{id}, g}"'] & X \times Y \arrow[r, "{\pi_0, h}"'] & X \times Y
\end{tikzcd}$$

We write $R_1\ g\ h\ j$ for this f.

Safe recursion differs from very safe recursion [Blo92] (Section 1.4.1) by using diagonal $\delta = \mathrm{id}, \mathrm{id}$ to repeatedly, rather than just once, read the parameters X. (The safe recursion in [BC92, Bel92] is actually closer to what we call dependent safe recursion in Chapter 4.)

Dyadic safe recursion [BC92] is that $\forall$ C commuting (for $k = 1,\ 2$)

$$\begin{tikzcd}
X \arrow[r, "g"] & Y & X \times Y \arrow[r, "h_k"] & Y & X \times T_2\, Y \arrow[d, shift left, "\pi_0"] \\
 & & & & X \arrow[u, shift left, "{\mathrm{id}, j}"]
\end{tikzcd}$$

∃! C commuting

$$\begin{tikzcd}
1 \times X \arrow[r, "0' \times X"] \arrow[d, "\pi_1"'] & N_1 \times X \arrow[r, "s_k \times X"] \arrow[d, "{\pi_1,\ f}"] & N_1 \times X \arrow[d, dashed, "{\pi_1,\ f}"] \\
X \arrow[r, "{\mathrm{id},\ g}"'] & X \times Y \arrow[r, "{\pi_0,\ h_k}"'] & X \times Y
\end{tikzcd}$$

We write $R_1'\ g\ h_1\ h_2\ j$ for this f.

Compatibility is that at tier 1 the $(0,\ s)$ and the $(0',\ s_1,\ s_2)$ can be defined in terms of each other using safe recursions. This is that

$$\begin{array}{ll}
G_1\ 0 = G_2\ 0' & \\
(G_1\ s) \circ (G_2\ s_1) = G_2\ s_2 & G_2\ (s_1 \circ 0') = G_1\ (s \circ 0) \\
(G_1\ s) \circ (G_2\ s_2) & (G_2\ s_1) \circ (G_1\ s) \\
= (G_2\ s_1) \circ (G_1\ s) & = (G_1\ (s \circ s)) \circ (G_2\ s_1)
\end{array}$$

We can give an essentially algebraic specification (Section 1.1.3) of the objects, and thus the maps, of 𝔓𝔖𝔭𝔞𝔠𝔢. Thus there exists an initial category **I** in 𝔓𝔖𝔭𝔞𝔠𝔢. Further (Appendix 1.B), **I** is the quotient of a Herbrand universe $\mathrm{colim}_i\ P_i$. We call the **I** maps *formal P space* maps and think of their representatives as programs. The *standard model* $\Gamma_V = (V \multimap \Gamma) \circ \chi$ of these formal maps is the composition

$$\begin{tikzcd}
\mathbf{I} \arrow[rr, "\Gamma_V"] \arrow[dr, "\chi"'] & & V \multimap \mathrm{set} \\
& V \multimap \mathbf{I} \arrow[ur, "V \multimap \Gamma"'] &
\end{tikzcd}$$

of the extent (Section 2.1.6) and the cotensor with 2 of the no inputs formal maps $\Gamma = \mathbf{I}(1,\ \_)$.

Proposition 2.2.1
For **I** *initial in the doctrine* 𝔓𝔖𝔭𝔞𝔠𝔢 *(as above)*

1. $\Gamma\ N_{0.1} = \{\mathrm{std}_{0.1}\ n \mid n \in N\}$, *where* $\mathrm{std}_{0.1}\ 0 = 0$, $\mathrm{std}_{0.1}\ (n+1) = s \circ \mathrm{std}_{0.1}\ n$.

2. $\Gamma\ N_{0.2} = \{\mathrm{std}_{0.2}\ n \mid n \in N\}$, *where* $\mathrm{std}_{0.2}\ 0 = 0'$, $\mathrm{std}_{0.2}\ (2n+k) = s_k \circ \mathrm{std}_{0.2}\ n$.

3. *Up to natural isomorphism, the unique* $\mathfrak{PSpace}$ *functor* $i : \mathbf{I} \to V \multimap \mathsf{set}$ *is* Γ_V.

Proof. Apply gluing to the comma category

$$\begin{tikzcd}
(V \multimap \mathsf{set})/\Gamma_V \arrow[r, "\pi_1"] \arrow[d, "\pi_0"'] \arrow[dr, phantom, "\Longrightarrow"] & \mathbf{I} \arrow[d, "\Gamma_V"] \\
V \multimap \mathsf{set} \arrow[r, "\mathrm{id}"'] & V \multimap \mathsf{set}
\end{tikzcd}$$

as in Appendix 1.F. As $N_{0.1}$, $N_{0.2}$ we take

$$\begin{tikzcd}
N \arrow[r] \arrow[d, "\mathrm{std}"] & 1 \arrow[d, "0 \mapsto \mathrm{id}"] & 1 \arrow[l] \arrow[d, "0 \mapsto \mathrm{id}"] \\
\Gamma\, N_1 \arrow[r] & \Gamma\, 1 & \Gamma\, 1 \arrow[l]
\end{tikzcd}
\qquad
\begin{tikzcd}
1 \arrow[r] \arrow[d, "0 \mapsto \mathrm{id}"] & 1 \arrow[d, "0 \mapsto \mathrm{id}"] & N \arrow[l] \arrow[d, "\mathrm{std}"] \\
\Gamma\, 1 \arrow[r] & \Gamma\, 1 & \Gamma\, N_1 \arrow[l]
\end{tikzcd}$$

where std makes sense because the unary and dyadic are compatible at tier 1. □

Proposition 2.2.2
The P space functions $N^I \to N^{I'}$ *are precisely those of the form* $\Gamma\, T_1\, T_2\, f$ *for* $\mathbf{I}$ *maps* f, *where* $\mathbf{I}$ *is initial in the doctrine* $\mathfrak{PSpace}$ *(as above).*

Proof. See Sections 2.3 and 2.4. □

2.3 Enough Maps

2.3.1 Getting Big

We will code (Section 2.3.2) machines (Section 1.4.3) inside I. To run these coded machines long enough, we need big enough I maps. By unary safe recursion we have addition

$$\begin{tikzcd}
1 \times N_{0.1} \arrow[r, "0 \times N_{0.1}"] \arrow[d, "\pi_1"'] & N_1 \times N_{0.1} \arrow[r, "s \times N_{0.1}"] \arrow[d, "{\pi_1, +}"] & N_1 \times N_{0.1} \arrow[d, "{\pi_1, +}"] \\
N_{0.1} \arrow[r, "{\mathrm{id}, \mathrm{id}}"'] & N_{0.1} \times N_{0.1} \arrow[r, "{\pi_0, s\, \pi_1}"'] & N_{0.1} \times N_{0.1}
\end{tikzcd}$$

and multiplication

$$\begin{tikzcd}
1 \times N_1 \arrow[r, "0 \times N_1"] \arrow[d, "\pi_1"'] & N_1 \times N_1 \arrow[r, "s \times N_1"] \arrow[d, "{\pi_1, \ast}"] & N_1 \times N_1 \arrow[d, "{\pi_1, \ast}"] \\
N_1 \arrow[r, "{\mathrm{id}, 0\, \tau}"'] & N_1 \times N_{0.1} \arrow[r, "{\pi_0, +}"'] & N_1 \times N_{0.1}
\end{tikzcd}$$

Analogously, by dyadic safe recursion we have concatenation

$$\begin{tikzcd}
1 \times N_{0.2} \arrow[r, "0' \times N_{0.2}"] \arrow[d, "\pi_1"'] & N_1 \times N_{0.2} \arrow[r, "s_k \times N_{0.2}"] \arrow[d, "{\pi_1, \bullet}"] & N_1 \times N_{0.2} \arrow[d, "{\pi_1, \bullet}"] \\
N_{0.2} \arrow[r, "{\mathrm{id}, \mathrm{id}}"'] & N_{0.2} \times N_{0.2} \arrow[r, "{\pi_0, s_k\, \pi_1}"'] & N_{0.2} \times N_{0.2}
\end{tikzcd}$$

and smash

$$\begin{tikzcd}
1 \times N_1 \arrow[r, "0' \times N_1"] \arrow[d, "\pi_1"'] & N_1 \times N_1 \arrow[r, "s_k \times N_1"] \arrow[d, "{\pi_1, \#}"] & N_1 \times N_1 \arrow[d, "{\pi_1, \#}"] \\
N_1 \arrow[r, "{\mathrm{id}, 0'\, \tau}"'] & N_1 \times N_{0.2} \arrow[r, "{\pi_0, \bullet}"'] & N_1 \times N_{0.2}
\end{tikzcd}$$

The I maps ${N_1}^I \to N_1$ built up from the standard numbers $s^n\, 0 : 1 \to N_1$ by $G_1\ +$, $G_1\ *$, $G_2\ \#$ are the *# polynomials*. Similarly we have # polynomials in set.

For $n \in N$ in set, n is the number of unary digits in n and we write $|n|$ for the number of dyadic digits in n. As

$$\sum_{i=0}^{|n|-1} 2^i k = (2^{|n|} - 1)k$$

we have that

$$|n| \leq log_2(n+1) \leq |n| + 1$$

Thus, given an N coefficient polynomial p in vector $|n|$ with components $|n_i|$, there are # polynomials q_0, q_1 such that

$$|q_0\, n| \leq p\, |n| \leq |q_1\, n|$$

E.g.

$$|n \# n| = |n|^2$$

Thus I initial in 𝔓𝔖𝔭𝔞𝔠𝔢 begins to look like [Tho72].

2.3.2 Coding Machines

We code dyadic register machines (Section 1.4.3) by unary safe recursions

$$\begin{tikzcd}
1 \times X \arrow[r, "0\times X"] \arrow[d, "\pi_1"'] & N_1 \times X \arrow[r, "s\times X"] \arrow[d, "\pi_1 . f"] & N_1 \times X \arrow[d, "\pi_1 . f"] \\
X \arrow[r, "\mathrm{id}, g"'] & X \times Y \arrow[r, "\pi_0 . h"'] & X \times Y
\end{tikzcd}$$

with $X = {N_1}^I$ containing the inputs and the program and $Y = {N_{0.1}}^J$ containing the outputs and the instruction pointer and data registers. Thus, for $n : 1 \to N_1$, $x : 1 \to X$, $f\ (n,\ x)$ is the state at time n.

We code the space bound by a time function $t : X \to N_1$. Then the P space functions will have the form (modulo tupling inputs in and projecting outputs out) $\Gamma\ G_1\ f'$ for composition

$$\begin{tikzcd}
X \arrow[r, "\mathrm{id}, \mathrm{id}"] \arrow[rrr, bend right, "f'"'] & X \times X \arrow[r, "t\times X"] & N_1 \times X \arrow[r, "f"] & Y
\end{tikzcd}$$

g initializes and is coded e.g. using γ_1 and 0.

2.3.3 Next State

h (from Section 2.3.2) codes the next state transition and decomposes into components h_j which compute the next value $1 \to N_{0.1}$ of the instruction pointer or a data register. (We can assume the outputs will be in data registers.) We need to simulate the dyadic operators s_1, s_2, C, D (Section 1.4.3).

By unary flat recursion we have predecessor

$$\begin{tikzcd}
1 \arrow[r, "0"] \arrow[dr, "0"'] & N_{0.1} \arrow[r, "s"] \arrow[d, "P"'] \arrow[dr, "\mathrm{id}"'] & N_{0.1} \arrow[d, "P"] \\
 & N_{0.1} & N_{0.1}
\end{tikzcd}$$

and conditional on test for zero

$$\begin{tikzcd}
1 \times (N_{0.1} \times N_{0.1}) \arrow[r, "0\times\_"] \arrow[dr, "\pi_0\ \pi_1"'] & N_{0.1} \times (N_{0.1} \times N_{0.1}) \arrow[r, "s\times\_"] \arrow[d, "z"'] \arrow[dr, "\pi_1\ \pi_1"'] & N_{0.1} \times (N_{0.1} \times N_{0.1}) \arrow[d, "z"] \\
 & N_{0.1} & N_{0.1}
\end{tikzcd}$$

Thus we can simulate s_1, s_2, C, D using s, Z, P and unary safe recursion. We can run the unary safe recursion (loops) long enough as, given the space bound, we can compute time bounds from the inputs in $1 \to X$. (This is similar to t in Section 2.3.2. See Section 2.3.4.)

Now (similarly to Section 1.4.5) h_j looks at a constant amount of low end dyadic digits and then, depending on what it sees, modifies a constant amount of low end dyadic digits. h_j can do this as follows.

1. Use C, D, γ_1, and projections to read and decide.

2. Use s_1, s_2, D and projections to modify.

2.3.4 Long Enough

With the inputs $n_i : 1 \to N_1$ the components of a vector $n : 1 \to {N_1}^{I'}$, a space bound polynomial in the $|n_i|$ (implicitly using Proposition 2.2.1) implies a time bound $2^{p\,|n|}$ with p an N coefficient polynomial (and $|n|$ the vector with components $|n_i|$) and thus implies, by Section 2.3.1, a time bound $q\ n$ with q a # polynomial. (As is classic [BC94, BDG88], count the states and note that repeated states imply infinite loops.) Thus, by Section 2.3.1, we have the t of Section 2.3.2 as well as the Section 2.3.3 variant of t.

2.4 Not Too Many Maps

Safety [BC92], in the case of V-comprehensions, is that in **I** tier 0.1, 0.2 inputs ($1 \to N_{0.1}$, $1 \to N_{0.2}$) can not affect tier 1 outputs ($\to N_1$) and that tier 0.1 (0.2) inputs can not affect tier 0.2 (0.1) outputs. Safety follows by applying $T_1\ T_2$ and using the naturality of $(\eta_1\ T_2) \circ \eta_2$ and by applying T_1 (T_2) and using the naturality of η_1 (η_2). E.g. consider

$$\begin{tikzcd}
N_{0.1} \arrow[r, "f"] \arrow[d, "\eta_1\ N_{0.1}"'] & N_{0.2} \arrow[d, "\eta_1\ N_{0.2}=\mathrm{id}"] \\
1 \arrow[r, "T_1\ f"'] & N_{0.2}
\end{tikzcd}$$

We will compile **I** maps f (actually their Herbrand universe representatives) to dyadic register machine codes which compute $\ulcorner G_1\ f$ and $\ulcorner G_2\ f$. At the same time we will obtain space and time bounds. For the induction on the structure of the Herbrand universe of **I** (as in Chapter 1) we will also need unary and dyadic output bounds.

For **I** maps

$$f : {N_1}^{I} \times {N_{0.1}}^{J} \rightarrow N_{0.1}$$
$$f' : {N_1}^{I'} \times {N_{0.2}}^{J'} \rightarrow N_{0.2}$$
$$f'' : {N_1}^{I''} \rightarrow N_1$$

(which are enough by safety as above and tuples as below) and variables $x_i : 1 \rightarrow N_1$, $y_j : 1 \rightarrow N_{0.k}$ we will show that (dropping o's and ,'s)

$$\begin{array}{cc} \text{space}(f\ x\ y) \leq |q'\ x\ y| & f\ x\ y \leq q\ x + \max_j y_j \\ \text{time}(f'\ x\ y) \leq p'\ |x|\ |y| & |f'\ x\ y| \leq p\ |x| + \max_j |y_j| \\ \text{space}(f''\ x) \leq |q'''\ x| & f''\ x \leq q''\ x \end{array}$$

where the # polynomials q, q', q'', q''' and the N coefficient polynomials p, p' depend only on f, f', f''.

Unlike in Chapter 1, we do not separate the names and values of variables into separate registers. Further, we keep inputs and outputs in separate registers, rather than sometimes totally overlapping them, e.g. to compute identities by doing nothing. But here, unlike there, we have the time and space to make copies. By the way, we count outputs, but not inputs, in the space.

Tuples $f,\ g : X \times Y \rightarrow U \times V$ (which combine diagonal, symmetry, and tensor) simply add the space and time involved. As, by the inductive hypothesis,

$$\begin{array}{cc} \text{space}(f\ x\ y) \leq |q'_f\ x\ y| & \text{space}(g\ x\ y) \leq |q'_g\ x\ y| \\ \text{time}(f'\ x\ y) \leq p'_{f'}\ |x|\ |y| & \text{time}(g'\ x\ y) \leq p'_{g'}\ |x|\ |y| \end{array}$$

it follows, using that $|n| \leq \log_2(n+1)$, that

$$\text{space}(f\ x\ y,\ g\ x\ y) \leq |(1 + q'_f\ x\ y)(1 + q'_g\ x\ y)|$$
$$\text{time}(f'\ x\ y,\ g'\ x\ y) \leq p'_{f'}\ |x|\ |y| + p'_{g'}\ |x|\ |y|$$

G_1, G_2 work right. Indeed, from

$$f\ x\ y \leq q\ x + \max_j y_j$$
$$|f'\ x\ y| \leq p\ |x| + \max_j |y_j|$$

it follows that

$$G_1\ (f\ x\ y) \leq q\ x + \textstyle\sum_j y_j$$
$$|G_2\ (f'\ x\ y)| \leq p\ |x| + \textstyle\sum_j |y_j|$$

Among the base functions, the variables $y : 1 \to N_{0.k}$ satisfy

$$y \leq y \qquad |y| \leq |y|$$

and take, to copy from an input register to an output register, linear space and time. Further

$$0 \leq 0 \qquad s\ y \leq 1 + y$$

and 0 and s take linear space while

$$|0'| \leq 0 \qquad |s_k\ y| \leq 1 + |y|$$

and $0'$ and s_k take linear time.

The coercions η_i, γ_i affect typing but not space, time, or output size. However, G_i, γ_i, together with Section 2.3.1 and the fact that time bounds imply space bounds, do mean that we do not need to consider the case $f'' : N_1^{l''} \to N_1$. E.g. we have

$$\gamma_1\ N_{0.1} : N_1 \to N_{0.1} \qquad G_1\ ((\gamma_1\ N_{0.1}) \circ f'') = f''$$

Unary flat recursion decomposes into scalar unary flat recursions

$$\begin{tikzcd}
1 \times Y \arrow[r, "0 \times Y"] \arrow[d, "\pi_1"'] & N_0 \times Y \arrow[r, "s \times Y"] \arrow[d, "f"'] \arrow[rd, "h"'] & N_0 \times Y \arrow[d, "f"] \\
Y \arrow[r, "g"'] & N_0 & N_0
\end{tikzcd}$$

which can be solved by

$$f\ n\ y = \bar{Z}\ n\ (g\ y)\ (h\ (\bar{P}\ n)\ y)$$

using the predecessor P and the conditional on test for zero Z from Section 2.3.3. But

$$P\ y \leq y \qquad Z\ y_0\ y_1\ y_2 \leq \max_j y_j$$

and P and Z take linear space.

Consider, with $X = {N_1}^I$, $Y = {N_{0.1}}^J$, $X' = {N_1}^{I'}$, $Y' = {N_{0.1}}^{J'}$, the composition

$$X \times Y \xrightarrow{g,h} X' \times Y' \xrightarrow{f} N_{0.1}$$

As, by the induction hypothesis, we have the vector inequalities

$$\begin{gathered} g\ x \leq q_g\, x \\ h\ x\ y \leq q_h\, x + \max_j y_j \\ f\ x'\ y' \leq q_f\, x' + \max_{j'} y'_{j'} \end{gathered}$$

it follows that

$$f\ (g\ x)\ (h\ x\ y) \leq q_f\, q_g\, x + \sum_{j'} q_{h_{j'}}\, x + \max_j y_j$$

Further, as

$$\begin{gathered} \text{space}(g\ x) \leq |q'_g\, x| \\ \text{space}(h\ x\ y) \leq |q'_h\, x\ y| \\ \text{space}(f\ x'\ y') \leq |q'_f\, x'\ y'| \end{gathered}$$

we have, taking the input registers of f to be the output registers of g, h and using that $|n| \leq \log_2(n+1)$, that

$$\begin{aligned} &\text{space}(f\ (g\ x)\ (h\ x\ y)) \leq \\ &|(1 + q'_g\, x)(1 + q'_h\, x\ y)(1 + q'_f\, (q_g\, x)\ (q_h\, x + \textstyle\sum_j y_j))| \end{aligned}$$

Similarly consider, with $X = {N_1}^I$, $Y = {N_{0.2}}^J$, $X' = {N_1}^{I'}$, $Y' = {N_{0.2}}^{J'}$, the composition (where we now drop some primes)

$$X \times Y \xrightarrow{g,h} X' \times Y' \xrightarrow{f} N_{0.2}$$

As we have the vector inequalities

$$\begin{gathered} |g\ x| \leq p_g\, |x| \\ |h\ x\ y| \leq p_h\, |x| + \max_j |y_j| \\ |f\ x'\ y'| \leq p_f\, |x'| + \max_{j'} |y'_{j'}| \end{gathered}$$

it follows that

$$|f\ (g\ x)\ (h\ x\ y)| \leq p_f\, p_g\, |x| + \sum_{j'} p_{h_{j'}}\, |x| + \max_j |y_j|$$

Further, as

$$\begin{gathered} \mathrm{time}(g\ x) \leq p'_g\, |x| \\ \mathrm{time}(h\ x\ y) \leq p'_h\, |x|\ |y| \\ \mathrm{time}(f\ x'\ y') \leq p'_f\, |x'|\ |y'| \end{gathered}$$

we have, taking the input registers of f to be the output registers of g, h, that

$$\begin{aligned} &\mathrm{time}(f\ (g\ x)\ (h\ x\ y)) \leq \\ &p'_g\, |x| + p'_h\, |x|\ |y| + p'_f\, (p_g\, |x|)\ (p_h\, |x| + \textstyle\sum_j |y_j|) \end{aligned}$$

For safe recursions we need to consider **I** commuting

$$\begin{tikzcd} X \times T_k\, Y \arrow[r, shift left, "\pi_0"] & X \arrow[l, shift left, "\mathrm{id},\, j"] \end{tikzcd}$$

Applying $i : \mathbf{I} \to V \multimap \mathrm{set}$ and looking at the $0.k$ component, we have that $Y \approx {N_{0.k}}^J$ for some $J \in N$.

Thus consider, with $X = {N_1}^I$, $Y = {N_{0.2}}^J$, $Y' = {N_{0.2}}^{J'}$, the dyadic safe recursion

$$\begin{tikzcd} 1 \times (X \times Y) \arrow[r, "0 \times (X \times Y)"] \arrow[d, "\pi_1"'] & N_1 \times (X \times Y) \arrow[r, "s_k \times (X \times Y)"] \arrow[d, "\pi_1,\, f"] & N_1 \times (X \times Y) \arrow[d, "\pi_1,\, f"] \\ X \times Y \arrow[r, "\mathrm{id},\, g"'] & (X \times Y) \times Y' \arrow[r, "\pi_0,\, h_k"'] & (X \times Y) \times Y' \end{tikzcd}$$

We have the varying composition

$$f\ (s_{k_0}\, s_{k_1}\ \ldots\ 0)\ x\ y = h_{k_0}\ x\ y\ h_{k_1}\ x\ y\ \ldots\ g\ x\ y$$

As, by the inductive hypothesis, we have the vector inequalities

$$\begin{gathered} |g\ x\ y| \leq p_g\, |x| + \max_j |y_j| \\ |h_k\, x\ y\ y'| \leq p_{h_k}\, |x| + \max(\max_j |y_j|, \max_{j'} |y'_{j'}|) \end{gathered}$$

it follows that

$$|f\ n\ x\ y| \leq |n| \sum_{k,j'} p_{h_{k,j'}}\ |x| + \sum_{j'} p_{g_{j'}}\ |x| + \max_j |y_j|$$

Further, keeping in mind the time it takes to reverse n, control the loop, and move the $y'_{j'}$ to the inputs of the $h_{k,j'}$, we have that

$$\begin{aligned}
&\text{time}(f\ n\ x\ y) \leq \\
&|n|(A(|n| \textstyle\sum_{k,j'} p_{h_{k,j'}}\ |x| + \sum_{j'} p_{g_{j'}}\ |x| + \sum_j |y_j|) + B \\
&+ \textstyle\sum_{k,j'} p'_{h_{k,j'}}\ |x|\ |y|\ (|n| \sum_{k,j'} p_{h_{k,j'}}\ |x| + \sum_{j'} p_{g_{j'}}\ |x| + \sum_j |y_j|)) \\
&+ \textstyle\sum_{j'} p'_{g'_j}\ |x|\ |y| + C
\end{aligned}$$

with A, B, $C \in N$.

Finally consider, with $X = {N_1}^I$, $Y = {N_{0.1}}^J$, $Y' = {N_{0.1}}^{J'}$, the unary safe recursion

$$\begin{tikzcd}
1 \times (X \times Y) \arrow[r, "0\times(X\times Y)"] \arrow[d, "\pi_1"'] & N_1 \times (X \times Y) \arrow[r, "s\times(X\times Y)"] \arrow[d, "{\pi_1, f}"] & N_1 \times (X \times Y) \arrow[d, "{\pi_1, f}"] \\
X \times Y \arrow[r, "{\text{id}, g}"'] & (X \times Y) \times Y' \arrow[r, "{\pi_0, h}"'] & (X \times Y) \times Y'
\end{tikzcd}$$

We have the varying composition

$$f\ (s\ s\ \ldots\ 0)\ x\ y = h\ x\ y\ h\ x\ y\ \ldots\ g\ x\ y$$

As we have the vector inequalities

$$\begin{aligned}
g\ x\ y &\leq q_g\, x + \max\nolimits_j y_j \\
h\ x\ y\ y' &\leq q_h\, x + \max(\max\nolimits_j y_j, \max\nolimits_{j'} y'_{j'})
\end{aligned}$$

it follows that

$$f\ n\ x\ y \leq n \sum_{j'} q_{h_{j'}}\, x + \sum_{j'} q_{g_{j'}}\, x + \max_j y_j$$

Further, as we have that

$$\begin{aligned}
\text{space}(g\ x\ y) &\leq q'_g\, x\ y \\
\text{space}(h\ x\ y\ y') &\leq q'_h\, x\ y\ y'
\end{aligned}$$

we have, keeping in mind the space needed for the intermediates/outputs $y'_{j'}$ and using that $|n| \leq \log_2(n+1)$, that

$$\begin{aligned}
&\mathrm{space}(f\ n\ x\ y) \leq \\
&|(1 + n \textstyle\sum_{j'} q_{h_{j'}}\ x + \sum_{j'} q_{g_{j'}}\ x + \sum_j y_j)^{J'} \\
&(1 + \textstyle\sum_{j'} q'_{h_{j'}}\ x\ y\ (n \sum_{j'} q_{h_{j'}}\ x + \sum_{j'} q_{g_{j'}}\ x + \sum_j y_j)) \\
&(1 + \textstyle\sum_{j'} q'_{g_{j'}}\ x\ y)|
\end{aligned}$$

2.A Linear Space

The objects of the linear space doctrine $\mathfrak{LinSpace}$ (this descends from [Bel92, Rit63]) are

1. FP 2-comprehensions
$$(\mathbf{C},\ T,\ G,\ \eta,\ \epsilon)$$
2. with unary
$$1 \xrightarrow{0} N_0 \xrightarrow{s} N_0$$
in $\mathbf{C}$ such that $T\ N_0 = 1$ satisfying
3. unary flat recursion and
4. unary safe recursion.

The maps are those preserving witnessed structure and are thus strict.

As for $\mathfrak{LinTime}$ in Chapter 1, there exists an initial category $\mathbf{I}$ in $\mathfrak{LinSpace}$ and functors

$$\begin{tikzcd}
\mathbf{I} \arrow[rr, "\Gamma_2"] \arrow[dr, "\chi"'] & & 2 \multimap \mathrm{set} \\
& 2 \multimap \mathbf{I} \arrow[ur, "2\multimap\Gamma"'] &
\end{tikzcd}$$

with χ the extent and $\Gamma = \mathbf{I}(1,\ \_)$.

Proposition 2.A.1
For $\mathbf{I}$ initial in $\mathfrak{LinSpace}$ (as above)

1. *Up to natural isomorphism the unique* 𝔏𝔦𝔫𝔖𝔭𝔞𝔠𝔢 *functor* $i : \mathbf{I} \to 2 \multimap \mathbf{set}$ *is* Γ_2.

2. *The linear space functions* $N^I \to N^{I'}$ *are those of the form* $\Gamma\ T\ f$ *for* $\mathbf{I}$ *maps* f.

Proof. Modify the proofs for 𝔓𝔖𝔭𝔞𝔠𝔢. In Section 2.3, + and $*$ now run the machine long enough. In Section 2.4, replace # polynomials q by N coefficient polynomials p. □

2.B P Time

The objects of the P time doctrine 𝔓𝔗𝔦𝔪𝔢 (this descends from [BC92, Cob65]) are

1. FP 2-comprehensions
$$(\mathbf{C},\ T,\ G,\ \eta,\ \epsilon)$$

2. with dyadic
$$1 \xrightarrow{0} N_0 \xrightarrow{s_1} N_0 \xrightarrow{s_2} N_0$$
in $\mathbf{C}$ such that $T\ N_0 = 1$ satisfying

3. dyadic flat recursion as (in Chapter 1) and

4. dyadic safe recursion.

The maps are those preserving witnessed structure and are thus strict.

As for 𝔏𝔦𝔫𝔗𝔦𝔪𝔢 in Chapter 1, there exists an initial category $\mathbf{I}$ in 𝔓𝔗𝔦𝔪𝔢 and functors

$$\begin{tikzcd}
\mathbf{I} \arrow[rr, "\Gamma_2"] \arrow[dr, "\chi"'] & & 2 \multimap \mathbf{set} \\
& 2 \multimap \mathbf{I} \arrow[ur, "2 \multimap \Gamma"'] &
\end{tikzcd}$$

with χ the extent and $\Gamma = \mathbf{I}(1,\ \_)$.

Proposition 2.B.1
For $\mathbf{I}$ *initial in* 𝔓𝔗𝔦𝔪𝔢 *(as above)*

1. *Up to natural isomorphism the unique* $\mathfrak{PTime}$ *functor* $\imath : \mathbf{I} \to 2 \multimap$ **set** *is* Γ_2.

2. *The P time functions* $N^I \to N^{I'}$ *are those of the form* $\Gamma\ T\ f$ *for* **I** *maps* f.

Proof. Modify the proofs for $\mathfrak{PSpace}$. In Section 2.3, $\bullet$ and $\#$ now run the machine long enough. The initialization and next state are now coded as in Section 1.4.5 using dyadic flat recursion. In Section 2.4, check that dyadic flat recursion respects the output and time bounds. □

Chapter 3

Dependent Products and Church Numerals

Introduction

In Section 3.1 we use dependent product diagrams to study LCC (= locally cartesian closed) categories. In particular, we specify LCC categories using sketches and orthogonality, show Awodey's semantic version of Martin-Löf's axiom of choice, show that the Yoneda embedding is LCC, and recall how LCC functors generalize locally connected topological spaces. Although dependent product diagrams appear independently in [Ndj92], our dp stacking (Proposition 3.1.3.1) is from 1991.

Previously (in Chapters 1, 2) we have characterized complexity classes using SM and FP 2-comprehensions. In Section 3.2 we easily define FL 2-comprehensions, and with more work, define LCC 2-comprehensions. However, in Section 3.4, using ideas from [Lei94, LM92], and a little lambda calculus from Section 3.3, we find that Church numerals prevent LCC 2-comprehensions from characterizing complexity classes. We eventually hope to overcome this using a combination of comprehensions and fibrations.

As in Chapter 2, we write τ for terminal maps, and f, g for the tuple map of maps f and g having a common domain.

3.1 Dependent Products

3.1.1 Comma Objects

Given a map $f : X \to Y$ in a category $\mathbf{C}$, we have a dependent sum functor

$$\begin{aligned} \mathrm{ds}_f : \mathbf{C}/X &\to \mathbf{C}/Y \\ e &\mapsto f \circ e \end{aligned}$$

The pull-back functor is defined by the adjunction $\mathrm{ds}_f \dashv \mathrm{pb}_f$, which we sometimes write as $\Sigma_f \dashv f^*$, and is thus determined by terminal objects in the comma categories

$$\begin{tikzcd} \mathrm{ds}_f/g \arrow[r] \arrow[d, ""{name=L}] & 1 \arrow[d, "g", ""'{name=R}] \\ \mathbf{C}/X \arrow[r, "\mathrm{ds}_f"'] & \mathbf{C}/Y \arrow[Rightarrow, from=L, to=R] \end{tikzcd}$$

The terminal objects in the comma categories ds_f/g are just the pull-back diagrams

$$\begin{tikzcd} {} \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d, "g"] \\ {} \arrow[r, "f"'] & {} \end{tikzcd}$$

Thus a category $\mathbf{C}$ is *FL* (= having finite limits) iff it has terminal objects and all pull-back diagrams.

Now suppose that the category $\mathbf{C}$ is FL. Then the dependent product functor is defined by the adjunction $\mathrm{pb}_f \dashv \mathrm{dp}_f$, which we sometimes write as $f^* \dashv \Pi_f$, and is thus determined by the terminal objects in the comma categories

$$\begin{tikzcd} \mathrm{pb}_f/e \arrow[r] \arrow[d, ""{name=L}] & 1 \arrow[d, "e", ""'{name=R}] \\ \mathbf{C}/Y \arrow[r, "\mathrm{pb}_f"'] & \mathbf{C}/X \arrow[Rightarrow, from=L, to=R] \end{tikzcd}$$

These terminal objects are the *dependent product diagrams*

$$\begin{tikzcd}
 & \cdot \arrow[r] \arrow[dl, dashed] \arrow[d] \arrow[dr, phantom, "\mathrm{dp}", near start] & \cdot \arrow[d] \\
\cdot \arrow[r, "e"'] & \cdot \arrow[r, dashed, "f"'] & \cdot
\end{tikzcd}$$

(In the future, but not here, we will instead place the dp in the right upper corner, as suggested by A. Blass.) Thus a category is *LCC (= locally cartesian closed)* iff it is FL and has all dependent product diagrams.

In set, the category of small sets, any $K \to J \to I$ splits as indexed sets $J = \{J_i\}_{i\in I}$, and dependently indexed sets $K = \{K_{ji}\}_{j\in J_i,\, i\in I}$ with J_i the fiber over $i \in I$ and K_{ji} the fiber over $j \in J_i$. Then the dependent product diagram has the form

$$\begin{tikzcd}
 & \cdot \arrow[r] \arrow[dl] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}", near start] & \{\prod_{j\in J_i} K_{ji}\}_{i\in I} \arrow[d] \\
K \arrow[r] & J \arrow[r] & I
\end{tikzcd}$$

thus justifying the terminology 'dependent product'.

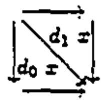
3.1.2 LCC Sketches

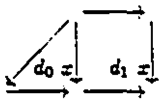
We define a sketch theory as (in Section 1.1) for *LCC sketches*. The sorts are

C_0	⇝ objects
C_1	⇝ maps
C_2	⇝ triangles
I	⇝ identities
L_0	⇝ terminals
L_2	⇝ pull-backs
P	⇝ dependent products

the operators are

$$\begin{array}{ll}
d_0\ x : C_0\ [x : C_1] & \rightsquigarrow \text{ codomain} \\
d_1\ x : C_0\ [x : C_1] & \rightsquigarrow \text{ domain} \\
d_0\ x : C_1\ [x : C_2] & \rightsquigarrow \text{ to map: } 1 \to 2 \\
d_1\ x : C_1\ [x : C_2] & \rightsquigarrow \text{ composite map: } 0 \to 2 \\
d_2\ x : C_1\ [x : C_2] & \rightsquigarrow \text{ from map: } 0 \to 1 \\
d\ x : C_1\ [x : I] & \\
d\ x : C_0\ [x : L_0] & \\
d_0\ x : C_2\ [x : L_2] & \rightsquigarrow \text{ as pictured below} \\
d_1\ x : C_2\ [x : L_2] & \\
d_0\ x : C_2\ [x : P] & \rightsquigarrow \text{ as pictured below} \\
d_1\ x : L_2\ [x : P] &
\end{array}$$





and the equations are

$$\begin{array}{l}
d_0\ d_0\ x = d_0\ d_1\ x : C_0\ [x : C_2] \\
d_1\ d_0\ x = d_0\ d_2\ x : C_0\ [x : C_2] \\
d_1\ d_1\ x = d_1\ d_2\ x : C_0\ [x : C_2] \\
d_0\ d\ x = d_1\ d\ x : C_0\ [x : I] \\
d_1\ d_0\ x = d_1\ d_1\ x : C_1\ [x : L_2] \\
d_1\ d_0\ x = d_2\ d_0\ d_1\ x : C_1\ [x : P]
\end{array}$$

The LCC categories with witnessed structure are then those LCC sketches orthogonal to the LCC sketch homomorphisms corresponding to the following basic almost equational assertions. We sometimes call such assertions entailments.

$\rightsquigarrow$ Associative composition

$$\begin{array}{l}
\{!\ f_1 \circ f_0 : C_2 \\
\quad d_0\ (f_1 \circ f_0) = f_1 : C_1 \\
\quad d_2\ (f_1 \circ f_0) = f_0 : C_1 \\
\quad [d_1\ f_1 = d_0\ f_0 : C_0\quad f_1 : C_1\quad f_0 : C_0]\}
\end{array}$$

$$\begin{array}{l}
\{d_1\, x_1 = d_1\, x_2 : C_1 \\
\;[d_0\, x_0 = d_0\, x_1 : C_1 \\
\quad d_1\, x_0 = d_0\, x_2 : C_1 \\
\quad d_2\, x_0 = d_0\, x_3 : C_1 \\
\quad d_2\, x_1 = d_1\, x_3 : C_1 \\
\quad d_2\, x_2 = d_2\, x_3 : C_1 \\
\quad x_0 : C_2 \quad x_1 : C_2 \quad x_2 : C_2 \quad x_3 : C_2]\}
\end{array}$$

⇝ Identity maps

$$\begin{array}{l}
\{!\ \mathrm{id}\ X : I \\
\quad d_0\, d\ \mathrm{id}\ X = X : C_0 \\
\quad [X : C_0]\} \\
\{d_1\, x = d_0\, x : C_1 \\
\;[d\, y = d_2\, x : C_1 \quad y : I \quad x : C_2]\} \\
\{d_1\, x = d_2\, x : C_1 \\
\;[d\, y = d_0\, x : C_1 \quad y : I \quad x : C_2]\}
\end{array}$$

⇝ Terminal objects

$$\begin{array}{l}
\{!\ 1 : L_0\} \\
\{!\ \tau\ X\ Y : C_1 \\
\quad d_0\, \tau\ X\ Y = d\ Y : C_0 \\
\quad d_1\, \tau\ X\ Y = X : C_0 \\
\quad [X : C_0 \quad Y : L_0]\}
\end{array}$$

⇝ Pull-backs, as pictured below

$$\begin{array}{l}
\{!\ f_0 \times f_1 : L_2 \\
\quad d_0\, d_0\, (f_0 \times f_1) = f_0 : C_1 \\
\quad d_0\, d_1\, (f_0 \times f_1) = f_1 : C_1 \\
\quad [d_0\, f_0 = d_0\, f_1 : C_0 \quad f_0 : C_1 \quad f_1 : C_1]\}
\end{array}$$

$\{!\ \Delta_0\ x_0\ x_1\ y : C_2$

$\quad \Delta_1\ x_0\ x_1\ y : C_2$

$\quad d_0\ \Delta_0\ x_0\ x_1\ y = d_2\ d_0\ y : C_1$

$\quad d_0\ \Delta_1\ x_0\ x_1\ y = d_2\ d_1\ y : C_1$

$\quad d_1\ \Delta_0\ x_0\ x_1\ y = d_2\ x_0 : C_1$

$\quad d_1\ \Delta_1\ x_0\ x_1\ y = d_2\ x_1 : C_1$

$\quad d_2\ \Delta_0\ x_0\ x_1\ y = d_2\ \Delta_1\ x_0\ x_1\ y : C_1$

$\quad [d_0\ x_0 = d_0\ d_0\ y : C_1$

$\quad\ d_0\ x_1 = d_0\ d_1\ y : C_1$

$\quad\ d_1\ x_0 = d_1\ x_1 : C_1$

$\quad\ x_0 : C_2 \quad x_1 : C_2 \quad y : L_2]\}$

$\rightsquigarrow$ Dependent products, as pictured below

$\{!\ \Pi\ f_1\ f_0 : P$

$\quad d_0\ d_0\ \Pi\ f_1\ f_0 = f_0 : C_1$

$\quad d_0\ d_0\ d_1\ \Pi\ f_1\ f_0 = f_1 : C_1$

$\quad [d_0\ f_0 = d_1\ f_1 : C_0 \quad f_1 : C_1 \quad f_0 : C_1]\}$

$\{!\ \Lambda_0\ x_0\ x_1\ y : C_2$

$\quad \Lambda_1\ x_0\ x_1\ y : L_2$

$\quad \Lambda_2\ x_0\ x_1\ y : C_2$

$\quad d_0\ \Lambda_0\ x_0\ x_1\ y = d_2\ d_0\ y : C_1$

$\quad d_1\ \Lambda_0\ x_0\ x_1\ y = d_2\ x_0 : C_1$

$\quad d_2\ \Lambda_0\ x_0\ x_1\ y = d_2\ d_0\ \Lambda_1\ x_0\ x_1\ y : C_1$

$\quad d_0\ d_0\ \Lambda_1\ x_0\ x_1\ y = d_2\ d_1\ d_1\ y : C_1$

$\quad d_0\ d_1\ \Lambda_1\ x_0\ x_1\ y = d_2\ \Lambda_2\ x_0\ x_1\ y : C_1$

$\quad d_2\ d_1\ \Lambda_1\ x_0\ x_1\ y = d_2\ d_1\ x_1 : C_1$

$\quad d_0\ \Lambda_2\ x_0\ x_1\ y = d_0\ d_1\ d_1\ y : C_1$

$\quad d_1\ \Lambda_2\ x_0\ x_1\ y = d_0\ d_1\ x_1 : C_1$

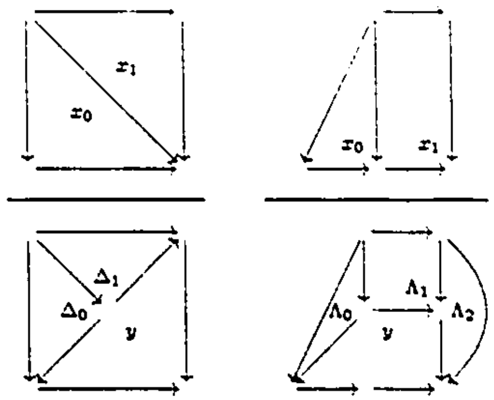
$\quad [d_0\ x_0 = d_0\ d_0\ y : C_1$

$\quad\ d_1\ x_0 = d_2\ d_0\ x_1 : C_1$

$\quad\ d_0\ d_0\ x_1 = d_0\ d_0\ d_1\ y : C_1$

$\quad\ x_0 : C_2 \quad x_1 : L_2 \quad y : P]\}$

Two of these entailments have the pictures



Thus, with $\mathbf{S}$ the above sketch theory of LCC sketches, and M the set of maps between finite LCC sketches which the above assertions amount to, $M \perp \mathbf{set}^{\mathbf{S}}$, the full subcategory of LCC sketches orthogonal to M, is the category of (small) LCC categories with witnessed structure, and strict LCC functors. More generally, a functor between LCC categories is LCC iff it preserves terminal objects, pull-backs, and dependent products (although not necessarily any choices of witnesses). Similarly one has FL (strict FL) functors between FL categories (with witnessed structure).

3.1.3 Pb and Dp Stacking

Given

$$\begin{tikzcd} & \mathbf{C} \arrow[d, shift right=2, "F"'] \arrow[d, phantom, "\dashv"] \\ 1 \arrow[r, "Y"'] & \mathbf{D} \arrow[u, shift right=2, "G"'] \end{tikzcd}$$

the counit $\epsilon_Y : F\,G\,Y \to Y$ is a terminal object in F/Y. Further, G on maps $g : Y_0 \to Y_1$ arises from the *stacking diagram*

$$\begin{tikzcd} F\,G\,Y_0 \arrow[r, "F\,G\,g"] \arrow[d, "\epsilon_{Y_0}"'] & F\,G\,Y_1 \arrow[d, "\epsilon_{Y_1}"] \\ Y_0 \arrow[r, "g"'] & Y_1 \end{tikzcd}$$

In particular, given an FL category C with a map $f : X \to Y$ we have

$$\begin{tikzcd}
 & \mathsf{C}/X \arrow[d, dashed, shift right=2, "\mathsf{ds}_f"'] \arrow[d, phantom, "\dashv"] \\
1 \arrow[r, "g"'] & \mathsf{C}/Y \arrow[u, shift right=2, "\mathsf{pb}_f"']
\end{tikzcd}$$

Then the stacking diagram for

$$\begin{tikzcd}
Z_0 \arrow[r, "h"] \arrow[dr, "g_0"'] & Z_1 \arrow[d, "g_1"] \\
 & Y
\end{tikzcd}$$

is an outer pull-back decomposed as

$$\begin{tikzcd}
{} \arrow[r] \arrow[rr, bend left] \arrow[d, dashed] & {} \arrow[r] \arrow[d, "\mathsf{pb}"] & X \arrow[d] \\
Z_0 \arrow[r, "h"] \arrow[rr, bend right, "g_0"'] & Z_1 \arrow[r, "g_1"] & Y
\end{tikzcd}$$

Pb stacking computes the outer pull-back by saying that the left inner square is a pull-back.

Similarly, given an LCC category C with a map $f : X \to Y$ we have

$$\begin{tikzcd}
 & \mathsf{C}/Y \arrow[d, shift right=2, "\mathsf{pb}_f"'] \arrow[d, phantom, "\dashv"] \\
1 \arrow[r, "e"'] & \mathsf{C}/X \arrow[u, shift right=2, "\mathsf{dp}_f"']
\end{tikzcd}$$

Then the stacking diagram for

$$\begin{tikzcd}
W_0 \arrow[r, "d"] \arrow[dr, "e_0"'] & W_1 \arrow[d, "e_1"] \\
 & X
\end{tikzcd}$$

is an outer dependent product decomposed as

$$\begin{tikzcd}
 & & {} \arrow[r, dashed] \arrow[d, dashed] \arrow[dl, dashed] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d, dashed] \\
 & {} \arrow[r, dashed] \arrow[dl] \arrow[d, phantom, "\mathrm{pb}"] & {} \arrow[r, dashed] \arrow[d] \arrow[dl, dashed] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d, dashed] \\
W_0 \arrow[r, "d"'] & W_1 \arrow[r, dashed, "e_1"'] & X \arrow[r, "f"'] & Y
\end{tikzcd}$$

Dp stacking computes the outer dependent product by claiming that the upper triangle and pull-back form a dependent product.

Proposition 3.1.3.1
The outside of

$$\begin{tikzcd}
 & & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
 & {} \arrow[r] \arrow[dl] \arrow[d, phantom, "\mathrm{pb}"] & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
{} \arrow[r] & {} \arrow[r] & {} \arrow[r] & {}
\end{tikzcd}$$

is a dependent product.

Proof. With the names

$$\begin{tikzcd}
 & {} \arrow[r, "d'"] \arrow[dl] \arrow[d, phantom, "\mathrm{pb}"] & {} \arrow[r, "f'"] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
{} \arrow[r, "d"'] & {} \arrow[r, "e"'] & {} \arrow[r, "f"'] & {}
\end{tikzcd}$$

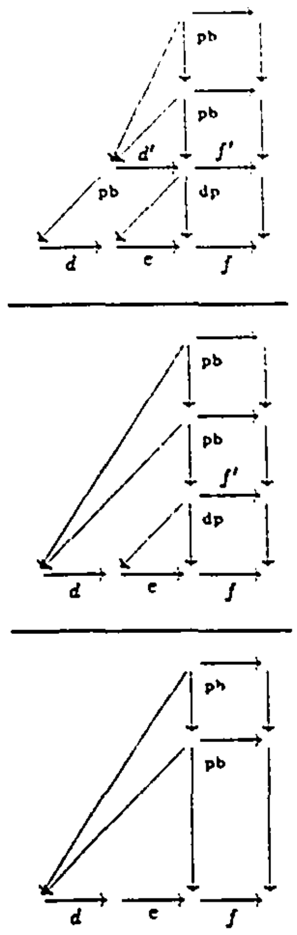
it is enough to show that the functor

$$F : \mathrm{pb}_{f'}/d' \to \mathrm{pb}_f/(e \circ d)$$

which on maps is

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d] \arrow[ddl] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] \\
 & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] \\
{} \arrow[r, "d'"'] & {} \arrow[r, "f'"'] & {}
\end{tikzcd}
\quad \mapsto \quad
\begin{tikzcd}
 & & {} \arrow[r] \arrow[d] \arrow[ddl] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] \\
 & & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] \\
 & {} \arrow[r, "d'"] \arrow[dl] \arrow[d, phantom, "\mathrm{pb}"] & {} \arrow[r, dashed, "f'"] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
{} \arrow[r, "d"'] & {} \arrow[r, "e"'] & {} \arrow[r, "f"'] & {}
\end{tikzcd}$$

is an equivalence, as equivalences preserve and reflect terminal objects. But, as pb and dp diagrams are terminal comma objects, we have bijections of maps



Thus F is faithful full. Removing the top comma object, we see that F is surjective on objects. □

3.1.4 Martin-Löf Choice

As a special case of dp stacking (Section 3.1.3) we have Streicher's [Str92, Awo94] semantic form of Martin-Löf's axiom of choice.

Proposition 3.1.4.1
The outer comma object in

$$\begin{tikzcd}
 & & {} \arrow[r] \arrow[dl, "\psi'"'] \arrow[d, "\mathrm{dp}"] & {} \arrow[d, "\Pi_{\pi_0}\ \psi'"] \\
 & {} \arrow[r] \arrow[dl, "\mathrm{pb}"] & Y^X \times X \arrow[r, "\pi_0"] \arrow[dl] \arrow[d, "\mathrm{dp}"] & Y^X \arrow[d] \\
{} \arrow[r, "\psi"'] & Y \times X \arrow[r, "\pi_1"'] & X \arrow[r] & 1
\end{tikzcd}$$

is a dependent product. Thus the right hand composition is

$$\Pi\, x : X\ \Sigma\, y : Y\ \psi\ x\ y \approx \Sigma\, f : Y^X\ \Pi\, x : X\ \psi\ x\ f\ x$$

Proof. We view ψ as a type dependent on $Y \times X$ whose sections witness its truth [See84]. Thus we write $\Pi\ x : X\ \psi\ x\ f\ x$ for $\Pi_{\pi_0}\ \psi'$. (We reverse the arguments of ψ as in Section 3.3.) □

This can also be viewed as a Skolemization or as a distributive law, and is a basis for the extraction of programs from specifications as in Nuprl [C+86].

3.1.5 Pull-Back is LCC

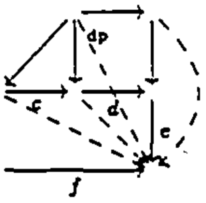
Consider a map $f : X \to Y$ in an LCC category **C**. By

$$\mathrm{ds}_f \dashv \mathrm{pb}_f \dashv \mathrm{dp}_f$$

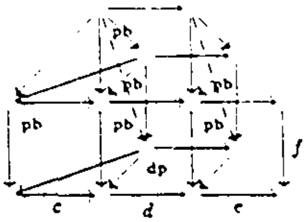
pb_f and dp_f are FL. Further

Proposition 3.1.5.1
pb_f *is LCC.*

Proof. As pb_f is FL, it remains to see that pb_f preserves dependent product diagrams. Start with



and get



in which a comma object over the pull-back of the bottom line uniquely maps to the pull-back of a dependent product over the bottom line. □

3.1.6 Presheaves

With **C** a category, the category of presheaves $\mathbf{set}^{\mathbf{C}^o}$ has object-wise FL structure. It also has LCC structure. Indeed, given

$$\begin{tikzcd} Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S \end{tikzcd}$$

in $\mathbf{set}^{\mathbf{C}^o}$, we describe the dependent product

$$\begin{tikzcd} & {} \arrow[ld, "@"'] \arrow[d] \arrow[r] \arrow[rd, phantom, "\mathrm{dp}"] & P \arrow[d, "\Pi_g f"] \\ Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S \end{tikzcd}$$

Think of **C** objects W as Joyal-Kripke worlds and **C** maps $l : W' \to W$ as localizations (or possible futures).

Proposition 3.1.6.1
With notation as above

1. *The elements of $P\ W$ (= P at world W) are the comma objects*

$$\begin{tikzcd} & {} \arrow[ld, "i"'] \arrow[d] \arrow[r] \arrow[rd, phantom, "\mathrm{pb}"] & y\ W \arrow[d, "j"] \\ Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S \end{tikzcd}$$

where $y : \mathbf{C} \to \mathbf{set}^{\mathbf{C}^o}$ is the Yoneda embedding $y\ X = \mathbf{C}(\_, X)$.

2. *$P\ l$ (for localization $l : W' \to W$) maps elements of $P\ W$ to elements of $P\ W'$ by commuting*

$$\begin{tikzcd}
& {} \arrow[r] \arrow[d] \arrow[ddl, dashed] \arrow[dr, phantom, "\mathsf{pb}"] & y\ W' \arrow[d, dashed, "y\ l"] \\
& {} \arrow[r] \arrow[d] \arrow[dl, dashed, "i"'] \arrow[dr, phantom, "\mathsf{pb}"] & y\ W \arrow[d, "j"] \\
Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S
\end{tikzcd}$$

3. *$\Pi_g\ f$ and @ are defined by*

$$\begin{aligned}
(\Pi_g\ f)_W\ (i,\ j) &= j_W\ \mathrm{id}_W \\
@_W\ (r,\ (i,\ j)) &= i_W\ (r,\ \mathrm{id}_W)
\end{aligned}$$

where $r \in R\ W$ is such that $g_W\ r = j_W\ \mathrm{id}_W$.

Proof. 1. and 2. As dependent products are terminal comma objects, and by the proof of the Yoneda lemma,

$$PW \approx \mathbf{set}^{\mathbf{C}^\circ}(y\ W,\ P)$$

with corresponding elements

$$\begin{array}{c}
\begin{tikzcd}
& {} \arrow[r] \arrow[d] \arrow[dl, "i"'] \arrow[dr, phantom, "\mathsf{pb}"] & y\ W \arrow[d, "j"] \\
Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S
\end{tikzcd} \\
\hline
\Lambda\ i\ j : y\ W \to P \\
\hline
(\Lambda\ i\ j)_W\ \mathrm{id}_W \in P\ W
\end{array}$$

3. We have commuting

$$\begin{tikzcd}
& {} \arrow[r] \arrow[d] \arrow[ddl, "i"'] \arrow[dr, phantom, "\mathsf{pb}"] & y\ W \arrow[d, "\Lambda\ i\ j"'] \arrow[dd, bend left, "j"] \\
& {} \arrow[r] \arrow[d] \arrow[dl, "@"] \arrow[dr, phantom, "\mathsf{dp}"] & P \arrow[d, "\Pi_g\ f"'] \\
Q \arrow[r, "f"'] & R \arrow[r, "g"'] & S
\end{tikzcd}$$

and thus commuting

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d] \arrow[ddl, dashed, "i_W"'] \arrow[dr, phantom, "\mathrm{pb}"] & \mathbf{C}(W, W) \arrow[d, dashed, "\Lambda\, i\, j"] & \ni \mathrm{id}_W \\
 & {} \arrow[r] \arrow[d] \arrow[dl, "\epsilon_W"] \arrow[dr, phantom, "\mathrm{pb}"] & P\,W \arrow[d, "(\Pi_g\, f)_W"] & \ni (\Lambda\, i\, j)_W\ \mathrm{id}_W \\
Q\,W \arrow[r, "f_W"'] & R\,W \arrow[r, "g_W"'] & S\,W & \ni j_W\ \mathrm{id}_W
\end{tikzcd}$$

□

3.1.7 Yoneda is LCC

We now have, using Proposition 3.1.6.1, an easy proof of the well known [Pit87] (A. Joyal circa 1974)

Proposition 3.1.7.1
The Yoneda embedding

$$\begin{aligned} y : \mathbf{C} &\to \mathbf{set}^{\mathbf{C}^\circ} \\ X &\mapsto \mathbf{C}(\_, X) \end{aligned}$$

is LCC.

Proof. That y is FL is essentially the definition of finite limits. Given

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d] \arrow[dl, "\epsilon"'] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d, "\Pi_g\, f"] \\
{} \arrow[r, "f"'] & {} \arrow[r, "g"'] & {}
\end{tikzcd}$$

in C, by the Yoneda lemma, the comma objects

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{pb}"] & y\,W \arrow[d] \\
{} \arrow[r, "y\,f"'] & {} \arrow[r, "y\,g"'] & {}
\end{tikzcd}$$

uniquely decompose as commuting

$$\begin{tikzcd}
 & {} \arrow[r] \arrow[d, dashed] \arrow[dl, phantom, "pb"] \arrow[ddl] & y\,W \arrow[d] \\
 & {} \arrow[r] \arrow[d] \arrow[dl, dashed, "y\,\mathfrak{A}"'] & {} \arrow[d, "y\,(\Pi_g\,f)"] \\
{} \arrow[r, "y\,f"'] & {} \arrow[r, "y\,g"'] & {}
\end{tikzcd}$$

□

3.1.8 Toposes and Locally Connected Maps

Although we will not need this below, it may be important elsewhere to know, as we recall here, that LCC functors generalize locally connected topological spaces.

Suppose that **C** is an FL category. Then a **C** map m is a monomorphism iff

$$\begin{tikzcd}
{} \arrow[r, "\mathrm{id}"] \arrow[d, "\mathrm{id}"'] & {} \arrow[d, "m"] \\
{} \arrow[r, "m"'] & {}
\end{tikzcd}$$

is a pull-back. Further, a monomorphism $\top$ is a *subobject classifier* iff $\forall$ monomorphisms m, $\exists!$

$$\begin{tikzcd}
{} \arrow[r] \arrow[d, "m"'] \arrow[dr, phantom, "pb"] & {} \arrow[d, "\top"] \\
{} \arrow[r] & {}
\end{tikzcd}$$

By a mildly tricky lemma [BW85], we can assume that $\top$ has the form

$$\top : 1 \to \Omega$$

By an easy lemma, any map $1 \to \Omega$ is a monomorphism.

A *topos* is an FL category **C** such that

1. **C** has a subobject classifier $\top : 1 \to \Omega$ and

2. $\forall$ C objects X, $\exists$

$$\begin{tikzcd}
 & \arrow[dl, dashed] \arrow[r, dashed] \arrow[d, dashed, "\mathrm{dp}"] & PX \arrow[d] \\
X \times \Omega \arrow[r, "\pi_0"'] & X \arrow[r] & 1
\end{tikzcd}$$

(PX is the *power* object.)

Thus we can specify toposes with witnessed structure using sketches and orthogonality (as in Section 3.1.2). Toposes are LCC [BW85].

A functor $F : \mathbf{E} \to \mathbf{S}$ between toposes is *geometric* iff it has an FL left adjoint and is *locally connected* iff it has an LCC left adjoint [Joh85]. When S is set, E is the sheaves of sets over a topological space X, and F is the global sections, X is locally connected iff F is.

3.2 LCC 2-Comprehensions

With $\mathbf{M} = \mathrm{end}(2)^{\circ}$ and $\mathfrak{LCC}$ the 2-category of LCC categories with witnessed structure, we will not define LCC 2-comprehensions to be 2-functors $\mathbf{M} \to \mathfrak{LCC}$, as

Proposition 3.2.1
The domain functor

$$d : \mathbf{set}^2 \to \mathbf{set}$$
$$(X : X_0 \to X_1) \mapsto X_0$$

is not LCC.

Proof. $\mathbf{set}^2$, $\mathbf{set}$ are LCC by Section 3.1.6. With $N = \{0,\ 1,\ 2,\ \dots\}$ the natural numbers in set, in $\mathbf{set}^2$ we have $N_0 = N \to 1$ and $N_1 = \mathrm{id} : N \to N$. Consider

$$\begin{tikzcd}
 & \arrow[dl] \arrow[r] \arrow[d, "\mathrm{dp}"] & {N_1}^{N_0} \arrow[d] \\
N_0 \times N_1 \arrow[r, "\pi_0"'] & N_0 \arrow[r] & 1
\end{tikzcd}$$

By Proposition 3.1.6.1, $d\,({N_1}^{N_0}) \approx$ the set of commuting

$$\begin{tikzcd}
 & {} \arrow[dl] \arrow[d, "\text{pb}"] \arrow[r] & 1 = 2(0.\ \_) \arrow[d] \\
N_0 \times N_1 \arrow[r, "\pi_0"'] & N_0 \arrow[r] & 1
\end{tikzcd}$$

$\approx \mathrm{set}^2(N_0,\ N_1) \approx$ the set of commuting

$$\begin{tikzcd}
N \arrow[r] \arrow[d] & N \arrow[d, "\text{id}"] \\
1 \arrow[r, "n"'] & N
\end{tikzcd}$$

$\approx N$. But $(d\,N_1)^{(d\,N_0)} = N^N$ is uncountable while N is not. □

Instead, we define *FL 2-comprehensions* to be 2-functors $\mathbf{M} \rightarrow \mathfrak{FL}$, where $\mathfrak{FL}$ is the 2-category of FL categories with witnessed structure, and *LCC 2-comprehensions* to be FL 2-comprehensions $(\mathbf{C},\ T,\ G,\ \eta,\ \epsilon)$, as in Chapters 1, 2, where $\mathbf{C}$ is LCC and the extent functor $\chi : \mathbf{C} \rightarrow \mathbf{C}^2$ (by $X \mapsto \chi\, X$ with $\chi = \eta \circ \epsilon$) is LCC. So we need, given an LCC category $\mathbf{C}$,

1. to show that $\mathbf{C}^2$ is LCC,

2. to describe the LCC structure on $\mathbf{C}^2$, and

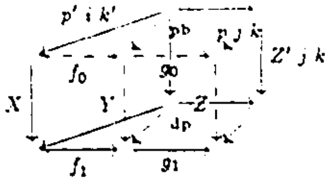
3. to show that $\mathbf{C}^2$ is canonically an LCC 2-comprehension.

(1. and 2. may be related to results in [Day70, Mak93].)

Proposition 3.2.2
Given an LCC category $\mathbf{C}$, $\mathbf{C}^2$ *is LCC. Indeed, over*

$$\begin{tikzcd}
{} \arrow[r, "f_0"] \arrow[d, "X"'] & {} \arrow[r, "g_0"] \arrow[d, "Y"'] & {} \arrow[d, "Z"'] \\
{} \arrow[r, "f_1"'] & {} \arrow[r, "g_1"'] & {}
\end{tikzcd}$$

we have the dependent product



where @, i, j, j', k, k', p, p', q are defined by

$$\begin{tikzcd}
{} & {} \arrow[r] \arrow[d] \arrow[dl, "k'"] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d, "k"] & & & \\
{} \arrow[r, "q"] & {} \arrow[r] \arrow[d, "j'"] \arrow[dl, "i"'] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d, "j"] & & & \\
{} \arrow[r] \arrow[d, "p'"'] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[r] \arrow[d] \arrow[rrr, bend left, "Y'"] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[r, "Z'"] \arrow[d, "p"'] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] & {} \arrow[l] \arrow[d] \arrow[dr, "@"] \arrow[dl, phantom, "\mathrm{dp}"] & {} \\
{} \arrow[r, "f_0"'] \arrow[rrrrr, bend right=60, "X"'] & {} \arrow[r, "g_0"'] \arrow[rrr, bend right, "Y"'] & {} \arrow[r, "Z"'] & {} & {} \arrow[l, "g_1"] & {} \arrow[l, "f_1"]
\end{tikzcd}$$

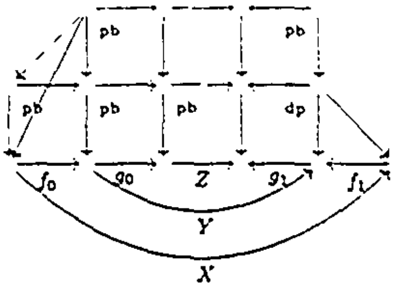
$$\begin{tikzcd}
{} \arrow[r, "q"] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d, "X\ p'\ i,\ @\ Y'\ j'"] \\
X_1 \arrow[r, "\mathrm{id},\mathrm{id}"'] & X_1
\end{tikzcd}$$

Proof. To verify that the above is a terminal comma object, we successively transform.

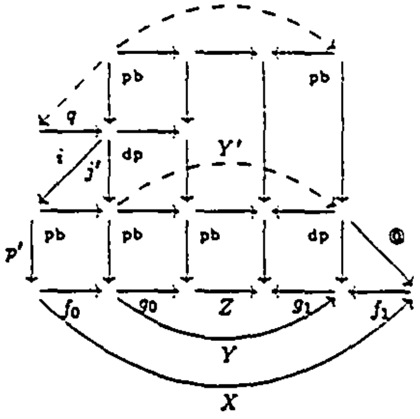
1.

$$\begin{tikzcd}
& {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[r] \arrow[d] & {} \arrow[d] & {} \arrow[l] \arrow[d] \arrow[dr] \arrow[dl, phantom, "\mathrm{pb}"] & \\
{} \arrow[r, "f_0"'] \arrow[rrrrr, bend right=60, "X"'] & {} \arrow[r, "g_0"'] \arrow[rrr, bend right, "Y"'] & {} \arrow[r, "Z"'] & {} & {} \arrow[l, "g_1"] & {} \arrow[l, "f_1"]
\end{tikzcd}$$

2.



3.



with

$$\begin{tikzcd}
{} \arrow[r, "q"] \arrow[d] \arrow[dr, phantom, "\mathsf{pb}"] & {} \arrow[d, "X\,p'\,i,\,@\,Y'\,j'"] \\
X_1 \arrow[r, "\mathrm{id},\mathrm{id}"'] & X_1
\end{tikzcd}$$

4.

$$\begin{tikzcd}
 & \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & \bullet \arrow[r] \arrow[d] & \bullet \arrow[ddd] \arrow[dr, phantom, "\mathrm{pb}"] & \bullet \arrow[l] \arrow[ddd] & \\
 & \bullet \arrow[r] \arrow[d] \arrow[dl, dashed, "q"'] \arrow[dr, phantom, "\mathrm{dp}"] & \bullet \arrow[d] & & & \\
\bullet \arrow[r] & \bullet \arrow[r] \arrow[d] \arrow[dl, "i"', "j'"] \arrow[dr, phantom, "\mathrm{dp}"] & \bullet \arrow[d] & & & \\
\bullet \arrow[r] \arrow[d, "p'"'] \arrow[dr, phantom, "\mathrm{pb}"] & \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & \bullet \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & \bullet \arrow[d] \arrow[dr, phantom, "\mathrm{dp}"] & \bullet \arrow[l] \arrow[d] \arrow[dr, "\alpha"] & \\
\bullet \arrow[r, "f_0"'] \arrow[rrrrr, bend right, "X"'] & \bullet \arrow[r, "g_0"'] \arrow[rrr, bend right, "Y"'] & \bullet \arrow[r, "Z"'] & \bullet & \bullet \arrow[l, "g_1"] & \bullet \arrow[l, "f_1"]
\end{tikzcd}$$

$\square$

Thus in particular

Proposition 3.2.3
Given an LCC category $\mathbf{C}$,

1. *the codomain functor*

$$c : \mathbf{C}^2 \to \mathbf{C}$$
$$(X : X_0 \to X_1) \mapsto X_1$$

and

2. *the identity functor*

$$\mathrm{id} : \mathbf{C} \to \mathbf{C}^2$$
$$X \mapsto (\mathrm{id} : X \to X)$$

are LCC.

Proof. 1. This is immediate from Proposition 3.2.2.

2. Directly, we have the bijections of comma objects

$$\begin{tikzcd}
 & & & {} \arrow[lll] \arrow[ld, dashed] \arrow[rr] \arrow[d, dashed] \arrow[d, phantom, "\mathrm{pb}", near start] & & {} \arrow[ld, dashed] \arrow[d, dashed] \\
{} \arrow[rr, dashed] \arrow[ddd, "\mathrm{id}"'] & & {} \arrow[rr, dashed] \arrow[ddd, dashed, "\mathrm{id}"'] & {} \arrow[rr, "\mathrm{id}"] \arrow[ddlll] \arrow[ddl] \arrow[ddr, phantom, "\mathrm{pb}"] & {} \arrow[l, phantom] & {} \arrow[ddl, dashed] \\
 & & & & & \\
{} & & & & {} \arrow[from=2-3, dashed, to=4-5, phantom] & \\
{} \arrow[rr, "f"'] & & {} \arrow[rr, "g"'] & & {} &
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[dd] & {} \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] & {} \arrow[dd] \\
 & {} \arrow[r] \arrow[d] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[d] & \\
{} & {} \arrow[r] \arrow[d] \arrow[dl] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] & {} \\
{} \arrow[r, "f"'] & {} \arrow[r, "g"'] & {} &
\end{tikzcd}$$

Alternatively, with a little effort, we can use Proposition 3.2.2. Consider

$$\begin{tikzcd}
 & & {} \arrow[r] \arrow[d] \arrow[dl, "q"] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
 & {} \arrow[r, "i"'] \arrow[dl, "i'"'] \arrow[d, phantom, "\mathrm{pb}"] & {} \arrow[r] \arrow[d] \arrow[dl, "f'"] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
{} \arrow[r, "{@,\ \mathrm{id}}"'] & {} \arrow[r] \arrow[d, "p'"'] \arrow[dr, phantom, "\mathrm{pb}"] & {} \arrow[r] \arrow[d] \arrow[dl, "@"] \arrow[dr, phantom, "\mathrm{dp}"] & {} \arrow[d] \\
 & {} \arrow[r, "f"'] & {} \arrow[r, "g"'] & {}
\end{tikzcd}$$

By dp stacking (Proposition 3.1.3.1) and f' (@, id) = id, it is enough to show that q is an equalizer of

$$\begin{tikzcd}
{} \arrow[r, shift left, "p'\,i"] \arrow[r, shift right, "@\,f'\,i"'] & {}
\end{tikzcd}$$

q so equalizes as $p'\ i\ q = p'\ (@,\ \mathrm{id})\ i' = @\ i' = @\ f'\ (@,\ \mathrm{id})\ i' = @\ f'\ i\ q$. Given q' such that $p'\,i\,q' = @\ f'\,i\,q'$, we have commuting

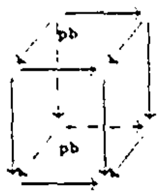
$$\begin{tikzcd}
 & {} \arrow[d, "q'"] \arrow[dl, "{f'\,i\,q',\,q'}"'] \\
{} \arrow[r, "q"'] & {}
\end{tikzcd}$$

if we can show that $(@,\ \mathrm{id})\ f'\ i\ q' = i\ q'$. But $p'\ (@,\ \mathrm{id})\ f'\ i\ q' = @\ f'\ i\ q'$ which $= p'\,i\,q'$ by assumption, while $f'\ (@,\ \mathrm{id})\ f'\,i\,q' = f'\,i\,q'$. □

Proposition 3.2.4
Given an LCC category **C**, $\mathbf{C}^2$ *is canonically an LCC 2-comprehension.*

Proof. With the FL structure

$$\begin{tikzcd}
1 \arrow[d] \\
1
\end{tikzcd}$$



on $\mathbf{C}^2$, the functors

$$T : \mathbf{C}^2 \to \mathbf{C}^2$$
$$(X : X_0 \to X_1) \mapsto (\mathrm{id} : X_0 \to X_1)$$

$$G : \mathbf{C}^2 \to \mathbf{C}^2$$
$$(X : X_0 \to X_1) \mapsto (\mathrm{id} : X_0 \to X_0)$$

are strict FL. The extent functor

$$\chi : \mathbf{C}^2 \to (\mathbf{C}^2)^2$$

takes

$$\begin{tikzcd}
X_0 \arrow[r, "f_0"] \arrow[d, "X"'] & Y_0 \arrow[d, "Y"] \\
X_1 \arrow[r, "f_1"'] & Y_1
\end{tikzcd}$$

to

$$\begin{tikzcd}
X_0 \arrow[rr, "f_0"] \arrow[dr, "X"'] \arrow[dd, "\mathrm{id}"'] & & Y_0 \arrow[dr, "Y"] \arrow[dd, dashed] & \\
& X_1 \arrow[rr, "\mathrm{id}", "f_1"'] \arrow[dd] & & Y_1 \arrow[dd, "\mathrm{id}"] \\
X_0 \arrow[rr, dashed, "f_0", "\mathrm{id}"'] \arrow[dr, "X"'] & & Y_0 \arrow[dr, dashed, "Y"] & \\
& X_1 \arrow[rr, "f_1"'] & & Y_1
\end{tikzcd}$$

Thus, since id : $\mathbf{C}^2 \to (\mathbf{C}^2)^2$ is LCC (by Proposition 3.2.3), so is χ (due to the symmetry $2 \times 2 \to 2 \times 2$). □

3.3 A Little Lambda Calculus

Suppose that **C** is an LCC category containing

$$1 \xrightarrow{0} N_0 \xrightarrow{s} N_0$$

We recall a little lambda calculus so that we can easily name some of the objects and maps in **C**.

A *type* is one of (by well-founded induction)

1. N_0

2. $Y \uparrow X$, which we sometimes write as Y^X, where X and Y are types.

The binary operation $\uparrow$ associates to the right. Thus $X \uparrow Y \uparrow Z$ is $X \uparrow (Y \uparrow Z)$. Type N_0 names the object N_0. Type Y^X names as pictured.

$$\begin{tikzcd}
 & Y^X \times X \arrow[r, "\pi_0"] \arrow[dl, "@, \pi_1"'] \arrow[d, "\pi_1"'] \arrow[dr, phantom, "\mathrm{dp}", near start] & Y^X \arrow[d] \\
Y \times X \arrow[r, "\pi_1"'] & X \arrow[r] & 1
\end{tikzcd}$$

We write

$$Y^{X_{n-1} \times \ldots \times X_1 \times X_0}$$

as sugar for

$$((Y^{X_{n-1}} \ldots)^{X_1})^{X_0}$$

A *context* is a square brackets enclosed set of *declarations* $x : X$ with x a variable unique within the context and X a type. A context

$$[x_0 : X_0 \quad x_1 : X_1 \quad \ldots]$$

names the product cone

$$\begin{tikzcd}
(\cdots \times X_1) \times X_0 \arrow[r] \arrow[rr, bend left, "x_j"] \arrow[d, "x_0"'] \arrow[dr, phantom, "\mathrm{pb}", near start] & (\cdots \times X_1) \arrow[r] \arrow[d] & X_j \\
X_0 \arrow[r] & 1 &
\end{tikzcd}$$

Here, unlike in the other chapters, we have reversed the order of the arguments, as abstraction will pop and push between stacks of arguments. In particular, $[x_0 : X_0]$ names

$$\begin{tikzcd}
X_0 \arrow[d, "x_0=\mathrm{id}"] \\
X_0
\end{tikzcd}$$

and $[\,]$ names 1.

A *lambda term* f of type X relative to a context C, all of which we write as $f : X\ C$, is one of

1. a variable x, if there is a declaration $x : X$ in the context C,

2. the constant 0, if X is N_0,

3. the constant s, if X is ${N_0}^{N_0}$,

4. an application $f\ g$, which we sometimes write as $f\ @\ g$, if $f : X^Y\ C$ and $g : Y\ C$,

5. an abstraction $[u : U]\ f$, if $f : V\ [u : U]\ C$ and X is V^U.

Here, when C is

$$[x_0 : X_0 \quad x_1 : X_1 \quad \ldots]$$

$[u : U]\ C$ is

$$[u : U \quad x_0 : X_0 \quad x_1 : X_1 \quad \ldots]$$

Application associates to the left. Thus $f\ g\ h = (f\ g)\ h$. With the object X named by the type X and the object C the limit of the product cone named by the context C, lambda terms $f : X\ C$ name maps $C \to X$ by

1. A variable names a projection, as pictured above, in the product cone named by the context C.

2. 0 names $0 \circ \tau : C \to N_0$.

3. s names the Curried map $\tilde{s} \circ \tau$ as pictured.

$$\begin{tikzcd}
 & & \cdot \arrow[r] \arrow[d, "\text{pb}"] & C \arrow[d, "\tau"] \\
 & & N_0 \arrow[r] \arrow[d, "\text{pb}"] \arrow[ddl, "{s, \text{id}}"'] & 1 \arrow[d, "\tilde{s}"] \\
 & & \cdot \arrow[r] \arrow[d, "\text{dp}"] \arrow[dl] & {N_0}^{N_0} \arrow[d] \\
 & N_0 \times N_0 \arrow[r, "\pi_1"'] & N_0 \arrow[r] & 1
\end{tikzcd}$$

4. Applications $f @ g$ name as pictured.

$$\begin{tikzcd}
C \arrow[r, "{f, g}"'] \arrow[rr, bend left, "f @ g"] & X^Y \times Y \arrow[r, "@"'] & X
\end{tikzcd}$$

5. Abstractions $[u : U]\ f$ name as pictured.

$$\begin{tikzcd}
 & C \times U \arrow[r] \arrow[d, "\text{pb}"] \arrow[ddl, "{f, \pi_1}"'] & C \arrow[d, "{[u{:}U]\ f}"] \\
 & \cdot \arrow[r] \arrow[d, "\text{dp}"] \arrow[dl] & X = U^V \arrow[d] \\
V \times U \arrow[r, "\pi_1"'] & U \arrow[r] & 1
\end{tikzcd}$$

We write

$$[u_0 : U_0 \quad u_1 : U_1 \quad \ldots]\ f$$

as sugar for

$$[u_0 : U_0]\ ([u_1 : U_1]\ \ldots\ f)$$

3.4 Church Numerals

Suppose that (**C**, T, G, η, ϵ) is an LCC 2-comprehension (as in Section 3.2) such that **C** contains

$$1 \xrightarrow{0} N_0 \xrightarrow{s} N_0$$

with $T\ N_0 = 1$ and satisfying unary safe recursion. (Compare this with the doctrine $\mathfrak{Lin}\mathfrak{Space}$ in Appendix 2.A.) Recall that unary safe recursion is that with

$$1 \xrightarrow{0} N_1 \xrightarrow{s} N_1 \qquad = \qquad G\,(1 \xrightarrow{0} N_0 \xrightarrow{s} N_0)$$

$\forall$ commuting

$$X \xrightarrow{g} Y \qquad X \times Y \xrightarrow{h} Y \qquad \begin{tikzcd} X \times T\,Y \arrow[d, shift left, "\pi_0"] \\ X \arrow[u, shift left, "{\mathrm{id},\, j}"] \end{tikzcd}$$

$\exists!$ commuting

$$\begin{tikzcd} X \times 1 \arrow[r, "X \times 0"] \arrow[d, "\pi_0"'] & X \times N_1 \arrow[r, "X \times s"] \arrow[d, "{\pi_0,\, f}"] & X \times N_1 \arrow[d, "{\pi_0,\, f}"] \\ X \arrow[r, "{\mathrm{id},\, g}"'] & X \times Y \arrow[r, "{\pi_0,\, h}"'] & X \times Y \end{tikzcd}$$

In set we have addition $+$, multiplication $*$, exponential $\uparrow$, and super-exponential $\Uparrow$ by

$$\begin{array}{cc} x + 0 = x & x + (s\,n) = s\,(x + n) \\ x * 0 = 0 & x * (s\,n) = x + (x * n) \\ x \uparrow 0 = s\,0 & x \uparrow (s\,n) = x * (x \uparrow n) \\ x \Uparrow 0 = s\,0 & x \Uparrow (s\,n) = x \uparrow (x \Uparrow n) \end{array}$$

We will use Church numerals to simulate these in **C**. Thus we will show that the numeric functions representable by **C** can grow too fast to allow them to form a complexity class.

Define 'Leivant tiers' of Church numerals C_i by

$$C_0 = N_0 \qquad C_{i+1} = C_i^{C_i \times C_i^{C_i \times C_i} \times C_i^{C_i}}$$

Define

$$1 \xrightarrow{0} C_i \xrightarrow{s} C_i$$

by, keeping argument reversal in mind,

$$0\,g\,h\,x = g\,x \qquad (s\,n)\,g\,h\,x = h\,x\,(n\,g\,h\,x)$$

Thus, given $g : C_i \to C_i$, $h : C_i \times C_i \to C_i$, we can solve for commuting

$$\begin{tikzcd}
C_i \times 1 \arrow[r, "C_i \times 0"] \arrow[d, "\pi_0"'] & C_i \times C_{i+1} \arrow[r, "C_i \times s"] \arrow[d, "\pi_0{,} f"] & C_i \times C_{i+1} \arrow[d, "\pi_0{,} f"] \\
C_i \arrow[r, "\mathrm{id}{,} g"'] & C_i \times C_i \arrow[r, "\pi_0{,} h"'] & C_i \times C_i
\end{tikzcd}$$

by, with $\tilde{f} : C_i^{C_{i+1} \times C_i}$, $\tilde{g} : C_i^{C_i}$, $\tilde{h} : C_i^{C_i \times C_i}$ the Curried forms of f, g, h,

$$\tilde{f}\, x\, n = n\, \tilde{g}\, \tilde{h}\, x$$

In particular, we can solve for commuting

$$\begin{tikzcd}
1 \arrow[r, "0"] \arrow[d] & C_{i+1} \arrow[r, "s"] \arrow[d, "\epsilon_i"] & C_{i+1} \arrow[d, "\epsilon_i"] \\
1 \arrow[r, "0"'] & C_i \arrow[r, "s"'] & C_i
\end{tikzcd}$$

by, with ϵ_i confused with its Curried form,

$$\epsilon_i\, n = n\, ([x : C_i]\, 0)\, ([z : C_i \quad y : C_i]\, s\, y)\, 0$$

Proposition 3.4.1

With assumptions as above, define

$$\begin{array}{ll}
g_i : C_i^{C_i} & h_i : C_i^{C_i \times C_i} \\
g_0\, x = x & f_{i+1} : C_i^{C_{i+1} \times C_i} \\
g_1\, x = 0 & h_0\, x\, y = s\, y \\
g_{i+2}\, x = s\, 0 & h_{i+1}\, x\, y\, g\, h\, x' = f_{i+1}\, (\epsilon_i\, x)\, y \\
 & f_{i+1}\, x\, n = n\, g_i\, h_i\, x
\end{array}$$

Then

1. $f_1\, (s^2\, 0)\, (s^n\, 0) = s^{n+2}\, 0$

2. $\epsilon_0\, (f_2\, (s^2\, 0)\, (s^n\, 0)) = s^{2n}\, 0$

3. $\epsilon_0\, (\epsilon_1\, (f_3\, (s^2\, 0)\, (s^n\, 0))) = s^{2 \uparrow n}\, 0$

4. $\epsilon_0\, (\epsilon_1\, (\epsilon_2\, (f_4\, (s^2\, 0)\, (s^n\, 0)))) = s^{2 \Uparrow n}\, 0$

Proof. Use induction on n.

$$\begin{aligned}
&f_1\ (s^2\ 0)\ 0 = s^2\ 0 = s^{0+2}\ 0.\\
&f_1\ (s^2\ 0)\ (s^{n+1}\ 0)\\
&= h_0\ (s^2\ 0)\ (f_1\ (s^2\ 0)\ (s^n\ 0)) = s\ (s^{n+2}\ 0) = s^{(n+1)+2}\ 0.
\end{aligned}$$

$$\begin{aligned}
&\epsilon_0\ (f_2\ (s^2\ 0)\ 0) = \epsilon_0\ 0 = 0 = s^{(2\ 0)}\ 0.\\
&\epsilon_0\ (f_2\ (s^2\ 0)\ (s^{n+1}\ 0))\\
&= \epsilon_0\ (h_1\ (s^2\ 0)\ (f_2\ (s^2\ 0)\ (s^n\ 0))) = f_1\ (s^2\ 0)\ (s^{2n}\ 0) = s^{2(n+1)}\ 0.
\end{aligned}$$

$$\begin{aligned}
&\epsilon_0\ (\epsilon_1\ (f_3\ (s^2\ 0)\ 0)) = \epsilon_0\ (\epsilon_1\ (s\ 0)) = s\ 0 = s^{2\uparrow 0}\ 0.\\
&\epsilon_0\ (\epsilon_1\ (f_3\ (s^2\ 0)\ (s^{n+1}\ 0)))\\
&= \epsilon_0\ (\epsilon_1\ (h_2\ (s^2\ 0)\ (f_3\ (s^2\ 0)\ (s^n\ 0))))\\
&= \epsilon_0\ (f_2\ (s^2\ 0)\ (s^{2\uparrow n}\ 0)) = s^{2\uparrow(n+1)}\ 0.
\end{aligned}$$

$$\begin{aligned}
&\epsilon_0\ (\epsilon_1\ (\epsilon_2\ (f_4\ (s^2\ 0)\ 0))) = \epsilon_0\ (\epsilon_1\ (\epsilon_2\ (s\ 0))) = s\ 0 = s^{2\Uparrow 0}\ 0.\\
&\epsilon_0\ (\epsilon_1\ (\epsilon_2\ (f_4\ (s^2\ 0)\ (s^{n+1}\ 0))))\\
&= \epsilon_0\ (\epsilon_1\ (\epsilon_2\ (h_3\ (s^2\ 0)\ (f_4\ (s^2\ 0)\ (s^n\ 0)))))\\
&= \epsilon_0\ (\epsilon_1\ (f_3\ (s^2\ 0)\ (s^{2\Uparrow n}\ 0))) = s^{2\Uparrow(n+1)}\ 0.
\end{aligned}$$

□

Now the $T\ C_i$ are terminal. Thus we can apply unary safe recursion to

$$\begin{tikzcd}
1 \arrow[r, "0"] & C_4 \arrow[r, "s"] & C_4
\end{tikzcd}$$

to get commuting

$$\begin{tikzcd}
1 \arrow[r, "0"] \arrow[d] & N_1 \arrow[r, "s"] \arrow[d, "\epsilon"] & N_1 \arrow[d, "\epsilon"] \\
1 \arrow[r, "0"'] & C_4 \arrow[r, "s"'] & C_4
\end{tikzcd}$$

Thus, by Proposition 3.4.1, the numeric functions representable by **C** can grow too fast to allow them to form a complexity class.

Chapter 4

3-Comprehensions and Kalmar Elementary

Introduction

[LM92] related tiers to the Grzegorczyk hierarchy (as in [Ros84]). In Appendix 2.A, we used FP 2-comprehensions to characterize the linear space functions. These form the second level of the Grzegorczyk hierarchy. (Thus many complexity classes are variants of the second level.) Here we use FP 3-comprehensions to characterize the Kalmar elementary functions (as in [Ros84]). These form the third level of the Grzegorczyk hierarchy. Thus we translate some of [LM92] to category theory. The third level seems to be needed to reason (deterministically) about, e.g. prove the consistency of, the second level [Ros84].

The basic idea is to pump up linear space to Kalmar elementary using $x \uparrow y = x^y$ essentially as we pumped up linear space to P space in Chapter 2 using $x \# y$. However, rather than use 2 tier 0's, one unary and one dyadic, compatibly joined at a single tier 1

$$\begin{tikzcd}
0.1 \arrow[dr] & & 0.2 \arrow[dl] \\
& 1 &
\end{tikzcd}$$

we use 3 linearly ordered tiers

$$\begin{tikzcd}
0 \arrow[d] \\
1 \arrow[d] \\
2
\end{tikzcd}$$

each of which may as well be unary, as numeric base and space versus time do not matter for Kalmar elementary functions. The partial orders V and 3 roughly describe how loops may nest. The partial order 3 leads to FP 3-comprehensions (Section 4.1).

Working out this basic idea will be slightly technical. We introduce the three doctrines $\mathfrak{K}$, $\mathfrak{E}$, and $\mathfrak{E}'$ (Section 4.2) as well as the complexity class E space. $\mathfrak{K}$ simply describes the Kalmar elementary maps. $\mathfrak{E}$ consists of FP 3-comprehensions with flat recursion (although it is not actually needed) and tier 1 and tier 2 safe recursions. $\mathfrak{E}'$ differs from $\mathfrak{E}$ by using dependent safe recursions (as below) rather than safe recursions. $\mathfrak{K}$ and $\mathfrak{E}'$ are clearly related whereas $\mathfrak{K}$ and $\mathfrak{E}$ are not. $\uparrow$ polynomials are built up from N in set using $+$, $*$, and $\uparrow$. The E space functions are those computable (on Turing machines) within space bounded by an $\uparrow$ polynomial. The images in set and set^3 of initial categories in $\mathfrak{K}$ and $\mathfrak{E}$ are big enough to include E space (Section 4.3), while the image in set^3 of an initial category in $\mathfrak{E}'$ is within E space (Section 4.4).

Much as safe recursion differs from very safe recursion (Section 1.4.1) by being able to read the parameters (X) more the once, namely during iteration, dependent safe recursion differs from safe recursion by being able to read the control (or 'time') variable (N_i) during iteration. Thus the vector (or simultaneous) safe recursion in [BC92, Bel92, LM92, Lei94] is essentially our dependent safe recursion, while our safe recursion has become (through its sufficiency and naturalness) independent of 'time'.

This chapter, together with Chapter 2, replaces [Ott94] which in turn replaced [Ott93]. We assume knowledge of Chapters 1 and 2.

4.1 3-Comprehensions

4.1.1 Comprehensions

We abstract from set^3 (which is the cotensor $3 \multimap$ set in $\mathfrak{FP}$ as below) and its 3 tiers of numbers

$$\begin{tikzcd}
N_0 \arrow[r, phantom, "="] & N \arrow[d] & N_1 \arrow[r, phantom, "="] & N \arrow[d, "\mathrm{id}"] & N_2 \arrow[r, phantom, "="] & N \arrow[d, "\mathrm{id}"] \\
 & 1 \arrow[d] & & N \arrow[d] & & N \arrow[d, "\mathrm{id}"] \\
 & 1 & & 1 & & N
\end{tikzcd}$$

Thus with $\mathrm{M}_3 = \mathrm{end}(3)^\circ$ the partially ordered monoid of endomorphisms of the partial order 3 with the 1-cells reversed (i.e. acting on the right, rather than the left, of 3), an *FP 3-comprehension* is a 2-functor $\mathrm{M}_3 \to \mathfrak{FP}$, where $\mathfrak{FP}$ is the 2-category of small FP categories with witnessed structure, strict FP functors, and natural transformations.

Now we present $\mathrm{M}_3 = \mathrm{end}(3)^\circ$ as a 2-category. As generators we choose the 1-cells (acting on the right of 3)

$$\begin{tikzcd}
T_0 \arrow[r, phantom, "="] & 0 \arrow[dr] & 0 & G_0 \arrow[r, phantom, "="] & 0 \arrow[r] & 0 \\
 & 1 \arrow[r] & 1 & & 1 \arrow[ur] & 1 \\
 & 2 \arrow[r] & 2 & & 2 \arrow[r] & 2 \\
T_1 \arrow[r, phantom, "="] & 0 \arrow[r] & 0 & G_1 \arrow[r, phantom, "="] & 0 \arrow[r] & 0 \\
 & 1 \arrow[dr] & 1 & & 1 \arrow[r] & 1 \\
 & 2 \arrow[r] & 2 & & 2 \arrow[ur] & 2
\end{tikzcd}$$

and the 2-cells (naming the element-wise partial order)

$$G_i \xrightarrow{\epsilon_i} \mathrm{id} \xrightarrow{\eta_i} T_i$$

We also define

$$\chi_i = \eta_i \circ \epsilon_i$$

The partial order underlying $\mathbf{M}_3$ is (generated by)

$$\begin{tikzcd}
G_1\,G_0 \arrow[d, "G_0\,G_1\,\epsilon_0"] & & & \\
G_0\,G_1 \arrow[r, "G_0\,\epsilon_1"'] \arrow[d, "\epsilon_0\,G_1"] & G_0 \arrow[d, "\epsilon_0"] & & \\
G_1 \arrow[r, "\epsilon_1"'] \arrow[d, "\eta_0\,G_1"] & \mathrm{id} \arrow[r, "\eta_1"'] \arrow[d, "\eta_0"] & T_1 \arrow[d, "T_1\,\eta_0"] & \\
T_0\,G_1 \arrow[r, "T_0\,\epsilon_1"'] & T_0 \arrow[r, "\eta_1\,T_0"'] & T_1\,T_0 \arrow[r, "T_1\,T_0\,\eta_1"'] & T_0\,T_1
\end{tikzcd}$$

We then choose as relations

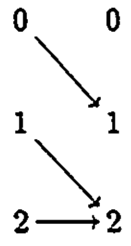
$$\begin{array}{cc}
G_i\,T_i = T_i & T_i\,\eta_i = \eta_i\,T_i = \mathrm{id} \\
T_i\,G_i = G_i & G_i\,\epsilon_i = \epsilon_i\,G_i = \mathrm{id} \\
 & T_i\,\epsilon_i = G_i\,\eta_i = \chi_i \\
T_1\,G_0 = T_1 & \epsilon_0\,T_1 = \eta_1\,G_0 = \eta_1 \circ \epsilon_0 \\
G_0\,T_1 = G_0 & \\
T_0\,G_1 = G_1\,T_0 & T_0\,\epsilon_1 = \epsilon_1\,T_0 \\
 & \eta_0\,G_1 = G_1\,\eta_0 \\
T_0\,T_1\,T_0 = T_0\,T_1 & \eta_0\,T_1\,T_0 = T_1\,T_0\,\eta_1 \\
G_1\,G_0\,G_1 = G_1\,G_0 & G_0\,G_1\,\epsilon_0 = \epsilon_1\,G_0\,G_1
\end{array}$$

In particular we have the adjunctions

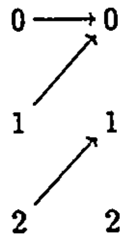
$$T_0 \dashv G_0 \dashv T_1 \dashv G_1$$

triples T_0, T_1, $T_0\,T_1$, and cotriples G_0, G_1, $G_1\,G_0$. We also have the non-idempotent ($(T_1\,T_0)^2 = T_0\,T_1$) modalities

$T_1\,T_0 =$



$G_0\,G_1 =$



Some of the 2-category assertions needed are not completely obvious. E.g. from

$$\begin{tikzcd}
\bullet & \bullet \arrow[l, bend right, "\mathrm{id}"', ""{name=A}] \arrow[l, bend left, "T_0", ""'{name=B}] & \bullet \arrow[l, "T_1"] & \bullet \arrow[l, bend right, "\mathrm{id}"', ""{name=C}] \arrow[l, bend left, "T_0", ""'{name=D}]
\arrow[Rightarrow, from=A, to=B, "\eta_0"]
\arrow[Rightarrow, from=C, to=D, "\eta_0"]
\end{tikzcd}$$

we have commuting

$$\begin{tikzcd}
T_1 \arrow[r, "T_1\,\eta_0"] \arrow[d, "\eta_0\,T_1"'] & T_1\,T_0 \arrow[d, "\eta_0\,T_1\,T_0"] \\
T_0\,T_1 \arrow[r, "T_0\,T_1\,\eta_0"'] & T_0\,T_1\,T_0
\end{tikzcd}$$

4.1.2 Extents

Suppose that $(\mathbf{C},\ T_i,\ G_i,\ \eta_i,\ \epsilon_i)$ is an FP 3-comprehension. We define an *extent* 2-natural transformation

$$\begin{tikzcd}
\mathbf{M}_3 \arrow[r, bend left=50, "\mathbf{C}", ""'{name=U}] \arrow[r, bend right=50, "3 \multimap \mathbf{C}"', ""{name=D}] & \mathfrak{FP}
\arrow[Rightarrow, from=U, to=D, "\chi"]
\end{tikzcd}$$

by the component, over the unique 0-cell of $\mathbf{M}_3$,

$$\begin{array}{ll}
\chi : \mathbf{C} \to 3 \multimap \mathbf{C} & \text{in } \mathfrak{FP} \\
\hline
3 \to \mathfrak{FP}(\mathbf{C},\ \mathbf{C}) & \text{in } \mathit{cat}
\end{array}$$

$$\begin{tikzcd}
0 \arrow[d] & & G_1\,G_0 \arrow[d, "G_1\,\chi_0"] \\
1 \arrow[d] & \mapsto & T_0\,G_1 \arrow[d, "T_0\,\chi_1"] \\
2 & & T_0\,T_1
\end{tikzcd}$$

Proposition 4.1.2.1
The extent functor χ is a 2-natural transformation between 3-comprehensions.

Proof. Use the multiplication table

	T_0	G_0	T_1	G_1
$G_1\,G_0$	$T_0\,G_1$	$G_1\,G_0$	$G_1\,G_0$	$G_1\,G_0$
$T_0\,G_1$	$T_0\,G_1$	$G_1\,G_0$	$T_0\,T_1$	$T_0\,G_1$
$T_0\,T_1$	$T_0\,T_1$	$T_0\,T_1$	$T_0\,T_1$	$T_0\,G_1$

and that the 2-celis are unique. □

4.1.3 Tiers

From

$$\begin{tikzcd}
N_0 & = & N \arrow[d] & N_1 & = & N \arrow[d, "\mathrm{id}"] & N_2 & = & N \arrow[d, "\mathrm{id}"] \\
 & & 1 \arrow[d] & & & N \arrow[d] & & & N \arrow[d, "\mathrm{id}"] \\
 & & 1 & & & 1 & & & N
\end{tikzcd}$$

in set^3 we abstract

$$T_0\,N_0 = 1 \qquad G_1\,N_0 = N_0 \qquad G_0\,N_0 = N_1 \qquad G_1\,N_1 = N_2$$

Proposition 4.1.3.1
For an FP 3-comprehension ($\mathbf{C}$, T_i, G_i, η_i, ϵ_i) *and objects N_i in* $\mathbf{C}$, *given the above equations, we have the multiplication table*

	N_0	N_1	N_2
T_0	1	N_1	N_2
G_0	N_1	N_1	N_2
T_1	N_0	N_0	N_2
G_1	N_0	N_2	N_2

□

4.2 Three Doctrines

4.2.1 $\mathfrak{K}$

$\mathfrak{K}$ objects consist of

1. FP categories $\mathbf{C}$
2. with

   $$1 \xrightarrow{0} N \xrightarrow{s} N$$

   in $\mathbf{C}$ and
3. $+$, Σ, $*$, Π, P, $-$ as below.

$\mathfrak{K}$ maps are the functors preserving witnessed structure and are thus strict.

$+$ satisfies

$$x + 0 = x \qquad x + (s\ n) = s\ (x + n)$$

which is that

$$\begin{tikzcd}
N \times 1 \arrow[r, "N\times 0"] \arrow[d, "\pi_0"'] & N \times N \arrow[r, "N\times s"] \arrow[d, "\pi_0{,}\,+"] & N \times N \arrow[d, "\pi_0{,}\,+"] \\
N \arrow[r, "\mathrm{id}{,}\,\mathrm{id}"'] & N \times N \arrow[r, "\pi_0{,}\,s\ \pi_1"'] & N \times N
\end{tikzcd}$$

commutes in $\mathbf{C}$.

Σ satisfies

$$(\Sigma\ f)\ x\ 0 = 0 \qquad (\Sigma\ f)\ x\ (s\ n) = f\ x\ n + (\Sigma\ f)\ x\ n$$

which is that, given $f : X \times N \to N$ in $\mathbf{C}$,

$$\begin{tikzcd}
X \times 1 \arrow[r, "X\times 0"] \arrow[dr, "(X\times 0){,}\,0\ \tau"'] & X \times N \arrow[r, "X\times s"] \arrow[d, "\mathrm{id}{,}\,\Sigma\ f"] & X \times N \arrow[d, "\mathrm{id}{,}\,\Sigma\ f"] \\
& (X \times N) \times N \arrow[r, "(X\times s)\ \pi_0{,}\,+\ (f\ \pi_0{,}\,\pi_1)"'] & (X \times N) \times N
\end{tikzcd}$$

commutes in $\mathbf{C}$.

$*$ satisfies

$$x * 0 = 0 \qquad x * (s\ n) = x + x * n$$

which is that

$$\begin{tikzcd}
N \times 1 \arrow[r, "N\times 0"] \arrow[d, "\pi_0"'] & N \times N \arrow[r, "N\times s"] \arrow[d, "{\pi_0, *}"] & N \times N \arrow[d, "{\pi_0, *}"] \\
N \arrow[r, "{\mathrm{id}, 0\top}"'] & N \times N \arrow[r, "{\pi_0, +}"'] & N \times N
\end{tikzcd}$$

commutes in **C**.

Π satisfies

$$(\Pi\ f)\ x\ 0 = s\ 0 \qquad (\Pi\ f)\ x\ (s\ n) = (f\ x\ n) * (\Pi\ f)\ x\ n$$

which is that, given $f : X \times N \to N$ in **C**,

$$\begin{tikzcd}
X \times 1 \arrow[r, "X\times 0"] \arrow[dr, "{(X\times 0), s\ 0\top}"'] & X \times N \arrow[r, "X\times s"] \arrow[d, "{\mathrm{id}, \Pi\ f}"] & X \times N \arrow[d, "{\mathrm{id}, \Pi\ f}"] \\
 & (X \times N) \times N \arrow[r, "{(X\times s)\ \pi_0, * (f\ \pi_0, \pi_1)}"'] & (X \times N) \times N
\end{tikzcd}$$

commutes in **C**.

P satisfies

$$P\ 0 = 0 \qquad P\ (s\ n) = n$$

which is that

$$\begin{tikzcd}
1 \arrow[r, "0"] \arrow[dr, "0"'] & N \arrow[r, "s"] \arrow[d, "P"'] \arrow[dr, "\mathrm{id}"'] & N \arrow[d, "P"] \\
 & N & N
\end{tikzcd}$$

commutes in **C**.

Finally $-$ satisfies

$$x - 0 = x \qquad x - (s\ n) = P\ (x - n)$$

which is that

$$\begin{tikzcd}
N \times 1 \arrow[r, "N\times 0"] \arrow[d, "\pi_0"'] & N \times N \arrow[r, "N\times s"] \arrow[d, "{\pi_0, -}"] & N \times N \arrow[d, "{\pi_0, -}"] \\
N \arrow[r, "{\mathrm{id}, \mathrm{id}}"'] & N \times N \arrow[r, "{\pi_0, P\ \pi_1}"'] & N \times N
\end{tikzcd}$$

commutes in **C**.

Proposition 4.2.1.1

1. *There is an initial category* **I** *in* $\mathfrak{K}$ *(as above).*

2. *Up to natural isomorphism, the unique* $\mathfrak{K}$ *functor* $i : \mathbf{I} \to \mathsf{set}$ *is* $\Gamma = \mathbf{I}(1, \_)$.

Proof.

1. Use almost equational specifications as in Section 1.1.
2. Use gluing as in Appendix 1.F. Thus consider the comma category

$$\begin{tikzcd}
\mathsf{set}/\Gamma \arrow[r, "\pi_1"] \arrow[d, "\pi_0"'] \arrow[rd, phantom, "\Longrightarrow"] & \mathbf{I} \arrow[d, "\Gamma"] \\
\mathsf{set} \arrow[r, "\mathrm{id}"'] & \mathsf{set}
\end{tikzcd}$$

We indicate some of the structure needed on set/Γ. As N take $\mathrm{std} : N \to \Gamma\, N$ where $\mathrm{std}\, 0 = 0$, $\mathrm{std}\,(s\, n) = s \circ \mathrm{std}\, n$. As

$$(\tilde{x} : \widetilde{X} \to \Gamma\, X) \times (\tilde{y} : \tilde{Y} \to \Gamma\, Y)$$

take

$$(x,\ y) \mapsto \tilde{x}\, x,\ \tilde{y}\, y : \widetilde{X} \times \tilde{Y} \to \Gamma\, (X \times Y)$$

As $+$ take

$$\begin{tikzcd}
N \times N \arrow[r, "+"] \arrow[d, "{(x,\, y) \mapsto \mathrm{std}\, x,\, \mathrm{std}\, y}"'] & N \arrow[d, "\mathrm{std}"] \\
\Gamma\, (N \times N) \arrow[r, "\Gamma +"'] & \Gamma\, N
\end{tikzcd}$$

□

4.2.2 $\mathfrak{C}$

$\mathfrak{C}$ objects consist of

1. FP 3-comprehensions $(\mathbf{C},\ T_i,\ G_i,\ \eta_i,\ \epsilon_i)$

2. with unary

$$1 \xrightarrow{0} N_0 \xrightarrow{s} N_0$$

in C such that

$$T_0\, N = 1 \qquad G_1\, N_0 = N_0$$

satisfying

3. unary flat recursion (as in Chapter 2) and

4. tier 1 and tier 2 unary safe recursion (as below).

$\mathfrak{C}$ maps are the functors preserving witnessed structure.

Define

$$\begin{aligned} 1 \xrightarrow{0} N_1 \xrightarrow{s} N_1 \quad &= \quad G_0\,(1 \xrightarrow{0} N_0 \xrightarrow{s} N_0) \\ 1 \xrightarrow{0} N_2 \xrightarrow{s} N_2 \quad &= \quad G_1\,(1 \xrightarrow{0} N_1 \xrightarrow{s} N_1) \end{aligned}$$

Notice that

$$\begin{array}{ll} T_0\, N_0 = 1 & T_0\, T_1\, N_0 = 1 \\ T_0\, N_1 = N_1 & T_0\, T_1\, N_1 = 1 \\ T_0\, N_2 = N_2 & T_0\, T_1\, N_2 = N_2 \end{array}$$

Thus *tier 1 unary safe recursion* is that ∀ C commuting

$$\begin{tikzcd} X \arrow[r, "g"] & Y & X \times Y \arrow[r, "h"] & Y & X \times T_0\, Y \arrow[d, shift left, "\pi_0"] \\ & & & & X \arrow[u, shift left, "{\mathrm{id}, j}"] \end{tikzcd}$$

∃! C commuting

$$\begin{tikzcd} X \times 1 \arrow[r, "X \times 0"] \arrow[d, "\pi_0"'] & X \times N_1 \arrow[r, "X \times s"] \arrow[d, "{\pi_0, f}"] & X \times N_1 \arrow[d, "{\pi_0, f}"] \\ X \arrow[r, "{\mathrm{id}, g}"'] & X \times Y \arrow[r, "{\pi_0, h}"'] & X \times Y \end{tikzcd}$$

while *tier 2 unary safe recursion* is that ∀ C commuting

$$\begin{tikzcd} X \arrow[r, "g"] & Y & X \times Y \arrow[r, "h"] & Y & X \times T_0\, T_1\, Y \arrow[d, shift left, "\pi_0"] \\ & & & & X \arrow[u, shift left, "{\mathrm{id}, j}"] \end{tikzcd}$$

∃! C commuting

$$\begin{tikzcd}
X \times 1 \arrow[r, "X\times 0"] \arrow[d, "\pi_0"'] & X \times N_2 \arrow[r, "X\times s"] \arrow[d, "{\pi_0, f}"] & X \times N_2 \arrow[d, "{\pi_0, f}"] \\
X \arrow[r, "{\mathrm{id}, g}"'] & X \times Y \arrow[r, "{\pi_0, h}"'] & X \times Y
\end{tikzcd}$$

With $\mathbf{I}$ initial in $\mathfrak{E}$, define Γ_3 by commuting

$$\begin{tikzcd}
\mathbf{I} \arrow[rr, "\Gamma_3"] \arrow[dr, "\chi"'] & & 3 \multimap \mathbf{set} \\
& 3 \multimap \mathbf{I} \arrow[ur, "3\multimap\Gamma"'] &
\end{tikzcd}$$

where χ is the extent (Section 4.1.2) and $\Gamma = \mathbf{I}(1, \_)$.

Proposition 4.2.2.1

1. *There exists an initial category* $\mathbf{I}$ *in* $\mathfrak{E}$ *(as above).*

2. *Up to natural isomorphism, the unique* $\mathfrak{E}$ *functor* $i : \mathbf{I} \to 3 \multimap \mathbf{set}$ *is* Γ_3.

Proof. Proceed as with Proposition 4.2.1.1, but with Γ_3 replacing Γ. □

4.2.3 $\mathfrak{E}'$

$\mathfrak{E}'$ differs from $\mathfrak{E}$ by replacing the safe recursions by dependent safe recursions. *Tier 1 unary dependent safe recursion* is that ∀ C commuting

$$X \xrightarrow{g} Y \qquad (X \times N_1) \times Y \xrightarrow{h} Y \qquad \begin{tikzcd}
(X \times N_1) \times T_0\, Y \arrow[d, shift left, "\pi_0"] \\
X \times N_1 \arrow[u, shift left, "{\mathrm{id}, j}"]
\end{tikzcd}$$

∃! C commuting

$$\begin{tikzcd}
X \times 1 \arrow[r, "X\times 0"] \arrow[dr, "{(X\times 0), g\,\pi_0}"'] & X \times N_1 \arrow[r, "X\times s"] \arrow[d, "{\mathrm{id}, f}"] & X \times N_1 \arrow[d, "{\mathrm{id}, f}"] \\
& (X \times N_1) \times Y \arrow[r, "{(X\times s)\,\pi_0, h}"'] & (X \times N_1) \times Y
\end{tikzcd}$$

while *tier 2 unary dependent safe recursion* is that ∀ **C** commuting

$$\begin{tikzcd}
X \arrow[r, "g"] & Y & (X \times N_2) \times Y \arrow[r, "h"] & Y & (X \times N_2) \times T_0\, T_1\, Y \arrow[d, shift left, "\pi_0"] \\
 & & & & X \times N_2 \arrow[u, shift left, "\mathrm{id},\, j"]
\end{tikzcd}$$

∃! **C** commuting

$$\begin{tikzcd}
X \times 1 \arrow[r, "X \times 0"] \arrow[dr, "(X \times 0),\, g\, \pi_0"'] & X \times N_2 \arrow[r, "X \times s"] \arrow[d, "\mathrm{id},\, f"] & X \times N_2 \arrow[d, "\mathrm{id},\, f"] \\
 & (X \times N_2) \times Y \arrow[r, "(X \times s)\, \pi_0,\, h"'] & (X \times N_2) \times Y
\end{tikzcd}$$

Proposition 4.2.3.1

1. *There exists an initial category* **I** *in* $\mathfrak{E}'$ *(as above).*

2. *Up to natural isomorphism the unique* $\mathfrak{E}'$ *functor* $i : \mathbf{I} \to 3 \multimap \mathbf{set}$ *is* Γ_3.

Proof. Proceed as with Proposition 4.2.2.1. □

The point of introducing $\mathfrak{E}'$ is that we have a functor

$$\mathfrak{E}' \to \mathfrak{K}$$
$$\mathbf{C} \mapsto \mathbf{C}_{(2)}$$

Given **C** in $\mathfrak{E}'$ let $\mathbf{C}_{(2)}$ be the full subcategory in **C** of X such that $T_0\, T_1\, X = X$. Define +, ∗, P as in Chapter 2 and then apply $G_1\, G_0$. Similarly define $\dot{-}$. Then define Σ, Π using tier 2 unary dependent safe recursion, $(\epsilon_1\, N_1) \circ f$, and +, ∗ with just G_0 applied.

We also have the underlying functor

$$\mathfrak{E}' \to \mathfrak{E}$$
$$\mathbf{C} \mapsto \mathbf{C}$$

4.2.4 E Space

† polynomials and E space were defined in the introduction.

Proposition 4.2.4.1
With $\mathbf{I}_{K}$, $\mathbf{I}_{E}$, $\mathbf{I}_{E'}$ initial in $\mathfrak{K}$, $\mathfrak{E}$, $\mathfrak{E}'$ (as above) the Kalmar elementary functions are precisely of the forms

1. *$\Gamma\ f$ for $\mathbf{I}_K$ maps f,*
2. *$\Gamma\ T_0\ T_1\ f$ for $\mathbf{I}_E$ maps f,*
3. *$\Gamma\ T_0\ T_1\ f$ for $\mathbf{I}_{E'}$ maps f,*
4. *the E space numeric functions.*

Proof. 1. This follows from Proposition 4.2.1.1.
2.–4. See sections 4.3 and 4.4. □

4.3 Enough Maps

Proposition 4.3.1
With $\mathbf{I}_K$, $\mathbf{I}_E$ initial in $\mathfrak{K}$, $\mathfrak{E}$ (as in Section 4.2) all the E space functions have the forms

1. *$\Gamma\ f$ for $\mathbf{I}_K$ maps f,*
2. *$\Gamma\ T_0\ T_1\ f$ for $\mathbf{I}_E$ maps f.*

Proof. Given an E space function, we run it on a dyadic register machine as in Chapters 1, 2 with an $\uparrow$ polynomial time bound. Then 1. follows by Theorem 3.1 in [Ros84]. In $\mathbf{I}_E$ we define $\uparrow$ such that

$$x \uparrow 0 = s\ 0 \qquad x \uparrow (s\ n) = x * (x \uparrow n)$$

by the tier 2 unary safe recursion

$$\begin{tikzcd}
N_1 \times 1 \arrow[r, "N_1 \times 0"] \arrow[d, "\pi_0"'] & N_1 \times N_2 \arrow[r, "N_1 \times s"] \arrow[d, "{\pi_0, \uparrow}"] & N_1 \times N_2 \arrow[d, "{\pi_0, \uparrow}"] \\
N_1 \arrow[r, "{id, s\, 0\, \tau}"'] & N_1 \times N_1 \arrow[r, "{\pi_0, G_0\, *}"'] & N_1 \times N_1
\end{tikzcd}$$

Thus $\uparrow$ polynomials are definable in $\mathbf{I}_E$. These allow, as in Chapter 2, running s_1, s_2, D, C simulations and dyadic register machines long enough. □

4.4 Not Too Many Maps

By Chapter 2 it remains to enumerate the Herbrand universe (as in Appendix 1.B) of $\mathbf{I}_{E'}$ initial in $\mathfrak{E}'$ (as in Section 4.2) and inductively establish unary output bounds.

Up to isomorphism, the objects of $\mathbf{I}_{E'}$ have the form

$$N_2{}^I \times N_1{}^J \times N_0{}^K$$

with I, J, $K \in N$. Due to G_i, ϵ_i and tuples, it is enough to consider maps

$$f : N_2{}^I \times N_1{}^J \times N_0{}^K \to N_0$$

With vectors $x : 1 \to N_2{}^I$, $y : 1 \to N_1{}^J$, $z : 1 \to N_0{}^K$, we will show that (modulo $\Gamma\ G_1\ G_0$)

$$f\ x\ y\ z \leq q\ x\ y + \max_k z_k$$

with the $\uparrow$ polynomials q (which depend only on f) not having y_j's within the right hand scopes of their $\uparrow$'s.

Set $e_i = \epsilon_i\ N_i$. Applying η_i, ϵ_i to N_j we get

$$\begin{array}{lll}
\eta_0\ N_0 = \tau & \eta_0\ N_1 = \mathrm{id} & \eta_0\ N_2 = \mathrm{id} \\
\epsilon_0\ N_0 = e_0 & \epsilon_0\ N_1 = \mathrm{id} & \epsilon_0\ N_2 = \mathrm{id} \\
\eta_1\ N_0 = \mathrm{id} & \eta_1\ N_1 = e_0 & \eta_1\ N_2 = \mathrm{id} \\
\epsilon_1\ N_0 = \mathrm{id} & \epsilon_1\ N_1 = e_1 & \epsilon_1\ N_2 = \mathrm{id}
\end{array}$$

Applying T_i, G_i to e_j we get

$$\begin{array}{cc}
T_0\ e_0 = \tau & T_0\ e_1 = e_1 \\
G_0\ e_0 = \mathrm{id} & G_0\ e_1 = e_1 \\
T_1\ e_0 = \mathrm{id} & T_1\ e_1 = e_0\ e_1 \\
G_1\ e_0 = e_0\ e_1 & G_1\ e_1 = \mathrm{id}
\end{array}$$

Safety, for 3-comprehensions, is that, for $j < i$, tier j inputs ($1 \to N_j$) can not affect tier i outputs ($\to N_i$). This safety follows (as in Chapter 2) by applying T_i and using the naturality of η_i.

We refine dependent safe recursion in $\mathbf{I}_{E'}$. By applying $i : \mathbf{I}_{E'} \to 3 \multimap \mathbf{set}$ and looking at the 0 component, $\mathbf{I}_{E'}$ commuting

$$\begin{tikzcd}(X \times N_1) \times T_0\, Y \arrow[r, shift left, "\pi_0"] & X \times N_1 \arrow[l, shift left, "\mathrm{id}, j"]\end{tikzcd}$$

implies that Y is isomorphic to some ${N_0}^K$. Similarly, $\mathbf{I}_{E'}$ commuting

$$\begin{tikzcd}(X \times N_2) \times T_0\, T_1\, Y \arrow[r, shift left, "\pi_0"] & X \times N_2 \arrow[l, shift left, "\mathrm{id}, j"]\end{tikzcd}$$

implies that Y is isomorphic to some ${N_1}^J \times {N_0}^K$. Thus it is enough to apply tier 1 dependent safe recursion to

$$g : X \to Y \qquad h : (X \times N_1) \times Y \to Y = {N_0}^K$$

and tier 2 dependent safe recursion to

$$g : X \to Y \qquad h : (X \times N_2) \times Y \to Y = {N_1}^J \times {N_0}^K$$

Using safety, tier 2 dependent safe recursion can be further refined by

1. obtain the ${N_1}^J$ part separately by tier 2 dependent safe recursion,
2. substitute the ${N_1}^J$ part away,
3. obtain the ${N_0}^K$ part using tier 1 dependent safe recursion.

Thus it is enough to apply tier 2 dependent safe recursion to

$$g : X \to Y \qquad h : (X \times N_2) \times Y \to Y = {N_1}^J$$

Most of the inductive cases are essentially the same as in Chapter 2. So consider the dependent safe recursions.

With $x : 1 \to X = {N_2}^I$, $y : 1 \to Y = {N_1}^J$, $z : 1 \to Z = {N_0}^K$, $n : 1 \to N_1$, $z' : 1 \to Z' = {N_0}^{K'}$, apply tier 1 dependent safe recursion to

$$g : X \times Y \times Z \to Z' \qquad h : X \times Y \times Z \times N_1 \times Z' \to Z'$$

Then

$$f\ x\ y\ z\ (s^n\ 0) = h\ x\ y\ z\ (s^{n-1}\ 0)\ h\ x\ y\ z\ (s^{n-2}\ 0)\ \ldots\ g\ x\ y\ z$$

Thus from the vector inequalities (from the inductive hypothesis)

$$\begin{array}{l} g\ x\ y\ z \leq q_g\ x\ y + \max_k z_k \\ h\ x\ y\ z\ n\ z' q_h\ x\ y\ n + \max(\max_k z_k, \max_{k'} z'_{k'}) \end{array}$$

we have

$$f\ x\ y\ z\ n \leq n(\sum_{k'} q_{h_{k'}}\ x\ y\ n) + \sum_{k'} q_{g_{k'}}\ x\ y + \max_k z_k$$

With $x : 1 \to X = {N_2}^I$, $y : 1 \to Y = {N_1}^J$, $n : 1 \to N_2$, $y' : 1 \to Y' = {N_1}^{J'}$, apply tier 2 dependent safe recursion to

$$g : X \times Y \to Y' \qquad h : X \times Y \times N_2 \times Y' \to Y'$$

(There are no N_0's due to safety.) Then

$$f\ x\ y\ (s^n\ 0) = h\ x\ y\ (s^{n-1}\ 0)\ h\ x\ y\ (s^{n-2}\ 0)\ \ldots\ g\ x\ y$$

Thus from

$$\begin{array}{l} g\ x\ y \leq q_g\ x\ y \\ h\ x\ y\ n\ y' \leq q_h\ x\ n\ y\ y' \\ \leq (q'_h\ x\ n + \sum_j y_j + \max_{j'} y'_{j'})^{q''_h\ x\ n} \end{array}$$

with $q'_{h_{j'}}$, $q''_{h_{j'}}$ independent of j', we have that

$$\begin{array}{l} f\ x\ y\ n \\ \\ \leq (q'_h\ x\ n + \sum_j y_j + (q'_h\ x\ n + \sum_j y_j +\ \ldots\ q_g\ x\ y\ \ldots)^{q''_h\ x\ n})^{q''_h\ x\ n} \\ \\ \leq (n(q'_h\ x\ n + \sum_j y_j) + q_g\ x\ y)^{(q''_h\ x\ n)^n} \end{array}$$

Bibliography

[AR94] J. Adámek and J. Rosický. *Locally Presentable and Accessible Categories*. Cambridge University Press, 1994.

[Awo94] S. Awodey. Axiom of choice and excluded middle in categorical logic. A talk at the Kansas City ASL meeting, 1994.

[BC92] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. In *STOC Proceedings*. ACM, 1992.

[BC94] D. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1994.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, 1988.

[BDG90] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*. Springer-Verlag, 1990.

[Bel92] S. Bellantoni. *Predicative Recursion and Computational Complexity*. PhD thesis, Department of Computer Science, University of Toronto, 1992.

[Blo92] S. Bloch. Alternating function classes within P. Technical Report 92-16, Department of Computer Science, University of Manitoba, 1992.

[Bor94] F. Borceux. *A Handbook of Categorical Algebra*. Cambridge University Press, 1994. Volumes 1–3.

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

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.

[C$^+$86] R. Constable et al. *Implementing Mathematics*. Prentice Hall, 1986.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Proceedings of the 1964 International Congress for Logic, Methodology, and the Philosophy of Science*. North-Holland, 1965.

[CRCM80] M. Coste-Roy, M. Coste, and L Mahé. Contribution to the study of the natural number object in elementary topoi. *Journal of Pure and Applied Algebra*, 17:35–68, 1980.

[Day70] B. Day. On closed categories of functors. In S. Mac Lane, editor, *Reports of the Midwest Category Seminar IV*, volume 137 of *LNM*. Springer-Verlag, 1970.

[FS90] P. Freyd and A. Scedrov. *Categories, Allegories*. North-Holland, 1990.

[GLT89] J. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[Hod93] W. Hodges. *Model Theory*. Cambridge University Press, 1993.

[HP93] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetic*. Springer-Verlag, 1993.

[Huw76] H. Huwig. *Beziehungen Zwischen Beschränkter Syntaktischer und Beschränkter Primitiver Rekursion.* PhD thesis, Informatik, Universität Dortmund, 1976.

[Huw82] H. Huwig. Ein modell des $P = NP$-problems mit einer positiven lösung. *Acta Informatica*, 17:221–243, 1982.

[Jay89] B. Jay. Languages for monoidal categories. *Journal of Pure and Applied Algebra*, 59:61–85, 1989.

[Jec78] T. Jech. *Set Theory.* Academic Press, 1978.

[JMS91] B. Jacobs, E. Moggi, and T. Streicher. Impredicative type theories. In D. Pitt et al., editors, *Category Theory and Computer Science*, volume 530 of *LNCS.* Springer-Verlag, 1991.

[Joh85] P. Johnstone. How general is a generalized space? In I. James and E. Kronheimer, editors, *Aspects of Topology.* Cambridge University Press, 1985.

[JS91] A. Joyal and R. Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88:55–112, 1991.

[Kel89] G. Kelly. Elementary observations on 2-categorical limits. *Bulletin of the Australian Mathematical Society*, 39:301–317, 1989.

[Kun80] K. Kunen. *Set Theory.* North-Holland, 1980.

[Law70] F. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. In A. Heller, editor, *Applications of Categorical Algebra.* AMS, 1970.

[Lei94] D. Leivant. Ramified recurrence and computational complexity I: Word algebras and poly-time. In P. Clote and J. Remmel, editors, *Feasible Mathematics II.* Birkhäuser, 1994.

[Llo87] J. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[LM92] D. Leivant and J. Marion. 1992. Unpublished notes.

[LM95] D. Leivant and J. Marion. Ramified recurrence and computational complexity II: Substitution and poly-space. Preprint, 1995.

[LS86] J. Lambek and P. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge University Press, 1986.

[Mak93] M. Makkai. The fibrational formulation of intuitionistic predicate logic I: Completeness according to Gödel, Kripke, and Läuchli, part 1. *Notre Dame Journal of Formal Logic*, 34:334–377, 1993.

[Mak94] M. Makkai. Generalized sketches as a framework for completeness theorems. Preprint, 1994.

[MP89] M. Makkai and R. Paré. *Accessible Categories*, volume 104 of *Contemporary Mathematics*. AMS, 1989.

[Ndj92] M. Ndjodo. *Systèmes de Réécriture et Cohérence des Isomorphismes de Types dans les Catégories Localement Closes*. PhD thesis, L'Universite d'Aix-Marseille II, Faculté des Sciences de Luminy, 1992.

[Ott93] J. Otto. Kalmar elementary and 2-simplices. Draft, 1993.

[Ott94] J. Otto. Kalmar, linear space, and P. Draft, 1994.

[P$^+$83] W. Paul et al. On determinism versus non-determinism and related problems. In *FOCS Proceedings*. IEEE Computer Society, 1983.

[P$^+$91] D. Pitt et al. *Category Theory and Computer Science*. Springer-Verlag, 1985, 1987, 1989, 1991. LNCS volumes 240, 283, 389, 530.

[Pav90] D. Pavlović. *Predicates and Fibrations*. PhD thesis, Faculteit der Wiskunde en Informatica, Rijksuniversiteit Utrech, 1990.

[Pit87] A. Pitts. Polymorphism is set theoretic, constructively. In D. Pitt et al., editors, *Category Theory and Computer Science*, volume 283 of *LNCS*. Springer-Verlag, 1987.

[PR89] R. Paré and L. Román. Monoidal categories with natural numbers object. *Studia Logica*, XLVIII:361–376, 1989.

[Rit63] R. Ritchie. Classes of predictably computable functions. *Transactions of the AMS*, 106:137–173, 1963.

[Rom89] L. Román. Cartesian categories with natural numbers object. *Journal of Pure and Applied Algebra*, 58:267–278, 1989.

[Ros84] H. Rose. *Subrecursion: Functions and Hierarchies*. Oxford University Press, 1984.

[See84] R. Seely. Locally cartesian closed categories and type theory. *Mathematical Proceedings of the Cambridge Philosophical Society*, 95:33–48, 1984.

[Str92] T. Streicher. Dependence and independence results for (impredicative) calculi of dependent types. *Mathematical Structures in Computer Science*, 2:29–54, 1992.

[Tho72] D. Thompson. Subrecursiveness: Machine-independent notions of computability in restricted time and storage. *Mathematical Systems Theory*, 6:3–15, 1972.

[Tro92] A. Troelstra. *Lectures on Linear Logic*. CLSI, 1992.

[Woo86] A. Woods. Bounded arithmetic formulas and Turing machines of constant alternation. In J. Paris, A. Wilkie, and G. Wilmers, editors, *Logic Colloquium '84*. North-Holland, 1986.

[Wra78] C. Wrathall. Rudimentary predicates and relative computation. *SIAM Journal on Computing*, 7:194–209, 1978.