Bachelor's Thesis

A Synthetic Proof of Myhill's Theorem in the Effective Topos

Author: Sara Rousta

Supervisor: Prof. Martin Hyland

Advisor: Dr. Anitha Thillaisundaram

Lund University

September 12, 2022

Abstract

One of the first results in classical computability theory was establishing the undecidability of the halting problem. In this thesis we will prove an even stronger version in the internal logic of the effective topos; more precisely in its full subcategory $Mod(\mathcal{K}_1)$ of modest sets internal to assemblies $Ass(\mathcal{K}_1)$. We will do this by proving that the diagonal halting set K is creative with our new definition. Our notion of creativity is classically equivalent to Post's and Myhill's definition, but importantly, it contains recursive content. The moral lesson is that if we do computability theory in the effective topos, the proofs turn out to be more constructive and in the spirit of what one intended to begin with.

Acknowledgements

First and foremost, I want to express my sincerest gratitude towards Professor Martin Hyland for introducing me to this subject and his guidance and patience throughout the entire process. I am thankful for our many enjoyable discussions and constructive meetings, which have influenced my mathematical taste and interest, and for teaching me that mathematical investigation can start with a 'why not'.

I also want to sincerely thank Dr. Anitha Thillaisundaram for giving me the opportunity to work on this topic, her kind and crucial support throughout the project and for arranging so many practical matters.

If it was not for Carina Geldhauser's mentorship I would not have applied for the internship out of which this project was engendered. I am thankful for all the encouragement and help.

The project was partially funded by the Philippa Fawcette Internship Programme and the Erasmus+ Traineeship Programme. I want to thank the people there for the financial support, which made the project as well as the many new friendships and beautiful memories possible. I am also thankful to the University of Cambridge for their hospitality.

To Matthias whose categorical support was my constant source of reassurence. Last but not least, I want to thank my family and friends. The number of occasions you were there for me cannot be effectively listed.

Contents

Introduction

The conclusion is inescapable that even for such a fixed, well-defined body of mathematical propositions, mathematical thinking is, and must remain, essentially creative.

Post

An analytic treatment of computability theory in a classical model for set theory inevitably leads to leaning heavily towards informal proof methods. They are of course partially justified by the empirical evidence provided by the works of Turing, Church and Kleene among others [8, 19, 22]. But informal methods are mainly used to avoid cumbersome details involving Gödel numbers to be able to get to the core mathematical ideas without having to deal with routine manipulations. This calls into question the appropriateness of the mathematical universe in which these ideas are encoded.

A suitable mathematical universe turns out to be Hyland's *effective topos* $\mathcal{E}ff$ [5]. Here, all functions are recursive or computable so that no reference to an external model of computation is necessary. Synthetic or axiomatic treatment of computability theory, pioneered by Bauer among others [1], allows us for instance to talk about recursively enumerable sets as just the (effective) sets, which are enumerable. In this sense, the synthetic approach reveals the mathematical structures without the encoded 'noise'. What is more, both the objects and morphisms between them carry constructive data in the effective topos. It therefore captures the essence of computability theory in which not only the results, but also the proofs are uniformly effective.

Our aim in this thesis is mainly to demonstrate how to translate results of computability theory in the classical world to the world of the effective topos. A certain amount knowledge of computability theory, logic and category theory is required to this end. To make the work self-contained, we avoided shortcuts and redid the proofs presented. In Chapter 1 we present two models of computation, discuss the encoding scheme and reproduce the classical result of the undecidability of the halting problem. Chapter 2 serves as a bridge to the effective topos where we also give an account of its internal intuitionistic logic. Finally, Chapter 3 culminates in a synthetic proof of Myhill's theorem, which is our main contribution. At the same time, the Curry-Howard-Lambek correspondence: formulae-as-objects, objects-as-types and proofs-as-morphisms, morphisms-as-algorithms is loosely explored.

Notation: The set of natural numbers $\mathbb{N}$ starts at 0. When the domain of comprehension is clear we simply write $\{ x \mid \dots \}$. We write $\overline{x}$ for a tuple of appropriate length $(x_1, \dots, x_n)$. The Kleene equality $\simeq$ used in the first chapter is defined in section 2.3. We will we will use λ-notation whenever convenient. The reader not familiar with λ-calculus can think of it as a prefix notation for 'maps to', viz. $\lambda x.f(x)$ can be taken to mean $x \mapsto f(x)$. See section 2.3 for a reference of untyped λ-calculus with partial application. Otherwise, we use standard notation that can be found in the references.

Chapter 1

Classical Computability Theory

This chapter will serve as a brief introduction to recursion theory or computability theory in the classical world, where the topics covered are only relevant to the chapters to come. The broad idea is to identify a class of (partial) functions that coincide with our intuition of an effective procedure: a deterministic finite procedure carried on in a discrete stepwise fashion with finite input and output. We study some properties of this class and establish the undecidability of the halting problem in a strong sense. The books by Roger [19, Chapters 1-7, 11] and Soare [22, Chapters 1-2] are our main references for this chapter.

1.1 Formal characterisations of algorithms

We begin by presenting Turing's characterisations of algorithms as it gives the most intuitive notion of an effective procedure carried out by a calculating agent with an infinite supply of ink and paper.

Definition 1.1.1. A *partial function* $A \rightharpoonup B$ is a pair (S, f) with a subset $S \subseteq A$ and a function $f\colon S \to B$.

A partial function generalises the notion of a function by allowing *at most* one output.

Definition 1.1.2 (Turing)**.** A Turing machine M consists of a two-way infinite *tape* divided into *cells* and a mechanical *reading head* containing a *Turing programme* P controlled by a partial map

$$\delta\colon Q \times S \to S \times D \times Q.$$

Here, $Q \equiv \{ q_0, q_1, \dots, q_n \}$, $n \geq 1$ denotes a finite set of *internal states* of the machine, $S \equiv \{ B, 1 \}$ (blank or 1) represents the symbols on each cell and $D \equiv \{ L, R \}$ (left or right) gives the direction in which the reading head moves.

We view δ as a finite set of quintuples (q, s, s', x, q') where machine M in state q

1. reads symbol s;
2. changes the symbol s to s';
3. moves the reading head one step either to $x = L$ or $x = R$;
4. switches internal state to q'.

Any finite set of quintuples may determine a programme P if every pair of quintuples differ in the first or second position. This *consistency criteria* restricts the machine in performing two or more courses of actions at once. The *input* $(x_1, x_2, \dots, x_k) \in \mathbb{N}^n$ are represented by $(x_i + 1), 1 \leq i \leq n$ consecutive 1's separated by a blank cell. The machine starts at the initial state q_1 on the leftmost 1 as shown in fig. 1.1. If the machine reaches

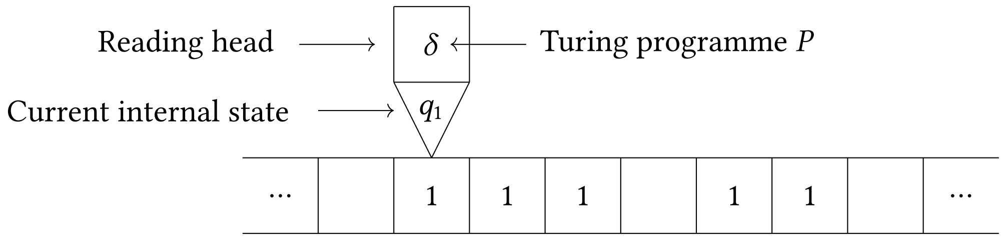

Figure 1.1: A Turing Machine M in an initital state with input $(2, 1)$.

the *halting state* q_0 after a finite number of steps, we say that M *halts* and the *output* integer is the number of consecutive 1's on the tape. A programme P determines a *partial computable function* $f\colon \mathbb{N}^n \rightharpoonup \mathbb{N}$, $n \in \mathbb{N}$ and we say that $f(x_1, \dots, x_n)\!\downarrow$ *converges* if machine M halts for programme P on input $(x_1, x_2, \dots, x_n) \in \mathbb{N}^n$. Otherwise, it is said to *diverge* $f(x_1, \dots, x_n)\!\uparrow$. A programme that determines an everywhere divergent function is given in fig. 1.2.

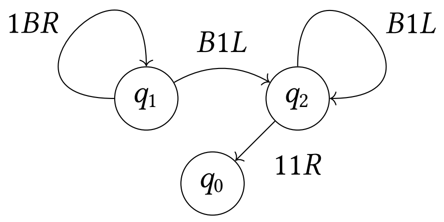

Figure 1.2: A program for the completely undefined function $f\colon \emptyset \rightharpoonup \mathbb{N}$.

Recursion is an integral component in many of the basic procedures we deem algorithmic such as factorial, exponentiation or sorting. Another way of formalising the notion of an effective procedure is in terms of recursion. We will obtain the class of partial recursive functions as a counterpart to the functions computable by a Turing machine.

Definition 1.1.3 (Gödel). The class of *primitive recursive functions* $\mathcal{C}$ is defined inductively as the least class of functions $\mathbb{N}^n \to \mathbb{N}, n \in \mathbb{N}$ containing

(i) the *constant functions* $C_c^n(x_1, \dots, x_n) \equiv c$ with $c, n \in \mathbb{N}$;

(ii) the *successor function* $S(x) \equiv x + 1$;

(iii) the *projections* $U_i^n(x_1, \dots, x_n) \equiv x_i, 1 \leq i \leq n$;

and closed under

(iv) *composition* such that for a k-ary function $h \in \mathcal{C}$, and n-ary functions $g_1, g_2, \dots, g_k \in \mathcal{C}$, the function

$$f(x_1, \dots, x_n) \equiv h(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$$

is in $\mathcal{C}$;

(v) *primitive recursion* such that for a $(n-1)$-ary function $g \in \mathcal{C}$, and $(n+1)$-ary function $h \in \mathcal{C}, n \geq 1$ the function f described by

$$\begin{aligned} f(0, x_2, \dots, x_n) &\equiv g(x_2, \dots, x_n), \\ f(x_1 + 1, x_2, \dots, x_n) &\equiv h(x_1, f(x_1, x_2, \dots, x_n), x_2, \dots, x_n) \end{aligned}$$

is in $\mathcal{C}$. (For $n = 1$, a 0-ary functions is taken to be constant and therefore in $\mathcal{C}$ by schema (i).)

Proposition 1.1.4. *Let $\mathcal{D}$ denote the class of functions for which there exists a sequence of functions $f_1, \dots, f_n$, with $f_n = f, n \in \mathbb{N}$ such that for each $i \leq n$, $f_i \in \mathcal{C}$ by (i) – (iii) or directly obtainable from some f_j, $j < i$ by (iv) or (v). Then $\mathcal{D}$ coincides with $\mathcal{C}$.*

Proof. The class $\mathcal{D}$ is closed under schemes (i) – (v) and contained in every class of functions closed under (i) – (v). ■

Such a sequence is called a *derivation* of a primitive recursive function f in $\mathcal{C}$. Consider for instance multiplication recursively defined as,

$$\begin{cases} M(0, y) \equiv 0; \\ M(x + 1, y) \equiv M(x, y) + y. \end{cases} \tag{1.1.1}$$

A derivation of multiplication could be as follows,

$$\begin{aligned}
&f_1(x) &&\equiv U_1^1(x) && \text{(iii)}\\
&f_2(x) &&\equiv S(x) && \text{(ii)}\\
&f_3(x_1, x_2, x_3) &&\equiv U_2^3(x_1, x_2, x_3) && \text{(iii)}\\
&f_4(x_1, x_2, x_3) &&\equiv f_2(f_3(x_1, x_2, x_3)) && \text{(iv)}\\
&f_5(0, x_2) &&\equiv f_1(x_2) && \\
&f_5(x_1 + 1, x_2) &&\equiv f_4(x_1, f_5(x_1, x_2), x_2) && \text{(v)}\\
&f_6(x_1, x_2, x_3) &&\equiv U_3^3(x_1, x_2, x_3) && \text{(iii)}\\
&f_7(x_1, x_2, x_3) &&\equiv f_5(f_3(x_1, x_2, x_3), f_6(x_1, x_2, x_3)) && \text{(iv)}\\
&f_8(x) &&\equiv 0 && \text{(i)}\\
&f_9(0, x_2) &&\equiv f_8(x_2) && \\
&f_9(x_1 + 1, x_2) &&\equiv f_7(x_1, f_9(x_1, x_2), x_2) && \text{(v)}\\
&f(x_1, x_2) &&\equiv f_9(x_1, x_2) &&
\end{aligned}$$

where the applied schema are indicated to the right. Note that $f_5(x_1, x_2) \equiv x_1 + x_2$, indeed there is a derivation of addition $f_1 \dots f_5$ as one would expect from eq. (1.1.1). Fig. 1.3 expresses f_5 equivalently as a Turing computable function. A list of basic primitive

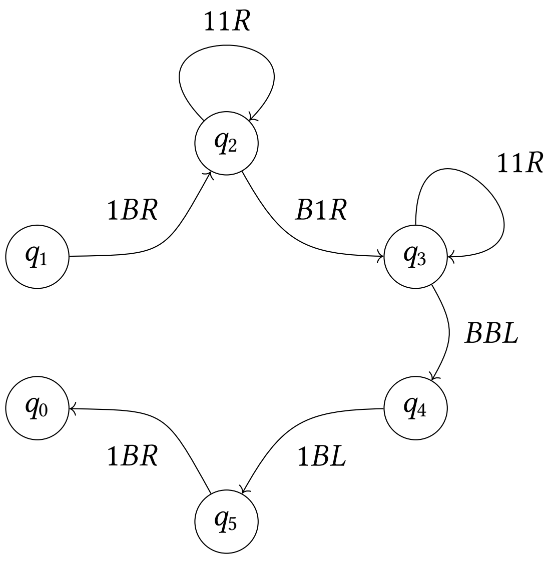

Figure 1.3: A program P for $\lambda xy.(x + y)$.

recursive functions can be found in [8, §44]. An important one is the prime enumeration function,

$$p(i) \equiv \text{the } (i+1)^{\text{th}} \text{ prime number.} \tag{1.1.2}$$

Let $p_0, p_1, \ldots, p_i, \ldots$ be an effective listing of the prime numbers. The fundamental theorem of arithmetic states that each $n \in \mathbb{N}$ has a unique representation,

$$n = p_0^{n_0} p_1^{n_1} \cdots p_i^{n_i} \cdots, \tag{1.1.3}$$

where all but finitely many n_i are zero. It follows that the prime factor representation eq. (1.1.3) is primitive recursive. The idea is to use the prime factor representation to assign a unique code to syntactical objects such as derivations of primitive recursive functions. For instance we assign to each symbol such as function symbols, parenthesis, variable symbols and numerals a unique natural number, which we then code up according to eq. (1.1.3). To give a concrete example, let us encode scheme (iii) independently in the following way:

U	x	$\circ$	$\_$	$\hat{}$	$($	$)$	$\equiv$	$,$
3	5	7	9	11	13	15	17	19

Then the instance $U_1^2(x_1, x_2) \equiv x_1$ becomes

U	$\circ\,\_$	$\circ\,\circ\,\hat{}$	$($	x	$,$	$\circ\, x$	$)$	$\equiv$	x
2^3	$3^{2^7 3^9}$	$5^{2^7 3^7 5^{11}}$	7^{13}	11^5	13^{19}	$17^{2^7 3^5}$	19^{15}	23^{17}	29^5

where $\circ$ works as a successor for the variable symbol x and digits for sub- and superscript numerals _ and ˆ. We can code up Turing programmes in a similar way. For more details see [19, §1.4] or [8, §52, §56]. For this encoding to be meaningful we would like it to be effectively invertible, viz. given a number we would want to retrieve the syntactical object it represents. It can be shown that the prime exponent function,

$$(n)_i \equiv \begin{cases} n_i & \text{if } n > 0; \\ 0 & \text{if } n = 0, \end{cases} \tag{1.1.4}$$

is primitive recursive for a fixed $i \geq 0$ [8, §45], which acts as a componentwise inverse to the numbering.

Definition 1.1.5. The above encoding scheme is known as *Gödel numbering* and we take it to be the canonical numbering of the theory.

Every primitive recursive function may be derived in a finite number of steps from an application of schema (i)–(v). Thus we could make a *listing*, viz. a surjective mapping from the set of natural numbers $\mathbb{N}$ to the class of primitive recursive functions $\mathcal{C}$ via the Gödel numbering. In principle we could device an algorithm that lists all primitive recursive functions of one variable. Suppose such a listing exists and let f_n denote the function determined by the $(n+1)^{\text{th}}$ derivation Q_n in this list. Now consider the function g defined as,

$$g(x) \equiv f_x(x) + 1. \tag{1.1.5}$$

Evidently, g is computable in the unrestricted sense: to compute $g(x)$, find Q_x in the list and retrieve the derivation to compute $f_x(x)$ and add one. However, g is not primitive recursive. For suppose towards a contradiction that $g = f_{x_0}$ for some $x_0 \in \mathbb{N}$, then $f_{x_0}(x_0) = g(x_0) \equiv f_{x_0}(x_0) + 1$, which is impossible by $S(x) \neq x$. This is an example of a proof by *Cantor's diagonalisation method.* The argument suggests that the class of primitive recursive functions is not exhaustive.

Definition 1.1.6 (Kleene)**.** The class C of *partial recursive* functions (*p.r*) is the least class closed under schema (i)–(v) of definition 1.1.3 and

(vi) *unbounded minimisation* such that for a partial recursive $(n+1)$-ary function $g \in C$, the function f described by

$$f(x_1, \dots, x_n) \equiv \mu y[g(x_1, \dots, x_n, y)\downarrow = 1 \quad \wedge \quad (\forall z < y)[g(x_1, \dots, x_n, z)\downarrow \neq 1]]$$

is in C.

Remark 1.1.7. In general, Kleene's μ-operator $\mu y R(x_1, \dots, x_n, y)$ gives the least y such that the predicate $R(x_1, \dots, x_n)$ holds or equivalently $\mu y[\chi(x_1, \dots, x_n, y) = 1]$, where χ is the *characteristic* or *representing function* of the predicate R,

$$\chi(x_1, \dots, x_n, y) \equiv \begin{cases} 1 & \text{if } R(x_1, \dots, x_n, y); \\ 0 & \text{otherwise.} \end{cases} \tag{1.1.6}$$

A predicate is said to be primitive recursive if it possesses a primitive recursive characteristic function. To give an example, the characteristic function of the equality predicate $x = y$ is given by $\overline{\text{sgn}}|x - y|$ and is primitive recursive [8, §45].

To see why we cannot repeat the same argument, fix a p.r function g and an input $\overline{x}$. Compute $g(\overline{x}, 0), g(\overline{x}, 1), \dots$ in order and do not proceed to the next unless $g(\overline{x}, z)$ converges. If there is a least y such that $g(\overline{x}, y)$ converges and is equal to 1, output y. Otherwise, proceed forever. Thus there may be inputs for which $f(x_1, \dots, x_n)$ diverges. We can therefore no longer diagonalise out of the class of partial recursive functions C as $f_{x_0}(x_0)$ may be undefined. The price we pay is that $\{ n \in \mathbb{N} \mid \phi_n \text{ is total} \}$ is no longer decidable. We make precise what this means in section 1.3.

Church's lambda calculus is yet another variant of formalisation for algorithmic procedures. What these models of computation have in common is content of the Church-Turing thesis. We summarise it here as fact.

Fact 1.1.8 (Church-Turing thesis)**.**

(i) The proposed formal characterisations of algorithms are shown to define the same class of partial functions and there exists a uniform effective procedure for translating a set of finite descriptions of one characterisation to another. For reference see for example [8, §68 – §69].

(ii) Partial functions that coincide with our intuitive notion of algorithmic functions are shown to have formal descriptions, which provides strong empirical evidence that this class of partial functions is sufficiently inclusive.

(iii) These characterisations are in the above sense the correct classification of our informal notion of algorithms.

The techniques developed in establishing the above points allow us to to give informal proofs, which we can think of as pseudo code that with some effort can be implemented in one of the models. This 'proof technique' is known as *proof by Church's Thesis* and asserts the validity of a proof independent of the degree of formality.

It is a remarkable fact that these characterisations are only equivalent up to the class of functions they define in the classical world. The different notions of computability turn out to give rise to two distinct realizability topoi, the world they model [7, 15].

In light of Church-Turing thesis the following definition is unambiguous.

Definition 1.1.9. Let P_e denote the set of instructions associated with with the code $e \in \mathbb{N}$ in the fixed listing of all sets of instructions. We call e the *index* or *Gödel number* of P_e. Let $\phi_e^{(n)}$ denote the partial recursive function of n variables determined by P_e. We call ϕ_e *total recursive* or simply *recursive* if it converges for every input.

Note that the Gödel numbering allows us to uniformly effectively go from an index to a programme and vice versa. The following is another coding scheme that is useful for encoding k-tuples.

Definition 1.1.10. Define a bijective primitive recursive coding $\langle - \rangle^{(k)} \colon \mathbb{N}^k \to \mathbb{N}, k \geq 0$ inductively as follows:

(i) $\langle - \rangle^{(1)} \equiv \lambda x.x$;

(ii) $\langle - \rangle^{(k+1)} \equiv \lambda x_1 \dots x_{k+1}.\langle \langle x_1, \dots, x_k \rangle^{(k)},\ x_{x+1} \rangle^{(2)}$,

where $\langle - \rangle^2 \colon \mathbb{N}^{(2)} \to \mathbb{N}$ is *Cantor's pairing function* $(x, y) \mapsto \frac{1}{2}(x + y)(x + y + 1) + x$. Let recursive functions $\pi_1^{(k)}, \dots, \pi_k^{(k)}$ denote the componentwise inverses of $\langle - \rangle^{(k)}$, that is for all z, $\langle \pi_1^{(k)}(z), \dots, \pi_k^{(k)}(z) \rangle^{(k)} = z$. The superscript is dropped whenever the context is clear.

Proposition 1.1.11. *Let ϕ be partial recursive functions of one variable, then*

(i) $\psi^{(k)} \equiv \lambda x_1 \dots x_k.\phi(\langle x_1, \dots, x_k \rangle)$

is a partial recursive functions of k variables. Let $\psi^{(k)}$ be a partial recursive function of k variables, then

(ii) $\phi \equiv \lambda z.\psi^{(k)}(\pi_1^{(k)}(z), \dots, \pi_k^{(k)}(z))$,

is a partial recursive function of one variable. Moreover, the association in (ii) is inverse to the association in (i).

Proof. Immediate by pairing in definition 1.1.10 and Church's Thesis. For given ϕ construct $\psi^{(k)}$ as in (i). Now define ϕ' as in (ii). Then

$$\phi'(z) = \psi^{(k)}(\pi_1^{(k)}(z), \dots, \pi_k^{(k)}(z)) = \phi(\langle \pi_1^{(k)}(z), \dots, \pi_k^{(k)}(z)\rangle) = \phi(z);$$

and similarly

$$\begin{aligned}\psi'(x_1, \dots, x_k) &= \phi(\langle x_1, \dots, x_k\rangle) \\ &= \psi^{(k)}(\pi_1^{(k)}(\langle x_1, \dots, x_k\rangle), \dots, \pi_k^{(k)}(\langle x_1, \dots, x_k\rangle)) \\ &= \psi^{(k)}(x_1, \dots, x_k).\end{aligned}$$ ■

1.2 Some standard results

We will formulate a few important results that will be used readily throughout the coming section.

Proposition 1.2.1. *There are exactly $\aleph_0$ partial recursive- and recursive functions.*

Proof. There are at least $\aleph_0$ recursive– and therefore partial recursive functions as all constant functions are recursive by Church's Thesis. By Gödel numbering there are at most $\aleph_0$ partial recursive– and therefore recursive functions. ■

Proposition 1.2.2. *There exist functions which are not recursive.*

Proof. Follows from diagonalisation. ■

The following result, makes precise the distinction between an effective procedure and mapping yielded by an effective procedure; a function may have multiple algorithms.

Lemma 1.2.3 (Padding lemma). *Each p.r function has $\aleph_0$ distinct indices.*

Proof. We give an informal proof. Let a p.r function ϕ_e be given. Let $\{ q_0, \dots, q_m \}$ be the internal states of the program P_e associated with ϕ_e. Add for each $k \in \mathbb{N}$ the extraneous quintuples $(q_{m+1}, 1, 1, R, q_{m+1}), \dots, (q_{m+k+1}, 1, 1, R, q_{m+k+1})$. The partial function determined by the new program $P_{e'}$ is unchanged as none of the states q_{m+k+1} can be entered. ■

Remark 1.2.4. The Padding lemma allows for the following useful construction. Define a recursive function t' such that

(i) $\lambda z.\phi_{t'(x,y)}(z) \simeq \lambda z.\phi_x(z)$;

(ii) $y \neq y' \implies t'(x,y) \neq t'(x,y')$,

by constructing successively larger Gödel numbers for the same partial recursive function. Then define a function t inductively as follows:

(i) $t(0,0) \equiv t'(0,0)$;

(ii) suppose $t(x',y')$ is defined for all $\langle x',\ y'\rangle < \langle x,\ y\rangle$, then set $t(x,y) \equiv t'(x,z)$,

where $z \equiv \mu w[(\forall\langle x',\ y'\rangle < \langle x,\ y\rangle)[t'(x,w) \neq t(x',y')]]$. Notice that t is also recursive as no two consecutive iterations of w can yield equality by construction of t'. That is to say, if $t'(x,w) = t(x',y')$ then it implies that $t'(x,w+1) \neq t(x',y')$ for a fixed (x',y'), hence z is always defined. Thus, we get the desirable properties

(i) $\lambda z.\phi_{t(x,y)}(z) \simeq \lambda z.\phi_x(z)$;

(ii) $[x \neq x' \text{ or } y \neq y'] \implies t(x,y) \neq t(x',y')$.

Theorem 1.2.5 (Kleene normal-form theorem)**.** *Fix $k \in \mathbb{N}$. There exist fixed primitive recursive functions p, t_k of one and $k+2$ variables respectively, such that for all e,*

$$\phi_e^{(k)} \simeq \lambda x_1 \dots x_k.p(\mu y[t_k(e,x_1,\dots,x_k,y) = 1]). \tag{1.2.1}$$

Proof. Define a function s of $k+3$ variables by,

$$s_k(e,\overline{x},y,s) \equiv \begin{cases} 1 & \text{if } P_e \text{ on input } \overline{x} \text{ yields an output y,} \\ & \text{in fewer than s steps of the computation;} \\ 0 & \text{otherwise.} \end{cases} \tag{1.2.2}$$

The encodings in the description are all primitive recursive and definition by cases for mutually exclusive predicates is primitive recursive, which gives that s is primitive recursive. Define two primitive recursive function $q \equiv \lambda y.(y+1)_0$ and $p \equiv \lambda y.(y+1)_1$. Finally let t_k be,

$$t_k \equiv \lambda e\overline{x}y.s_k(e,\overline{x},p(y),q(y)), \tag{1.2.3}$$

which is primitive recursive by definition 1.1.3. The theorem follows, for an even more formal proof see [8, §58]. ■

The next theorem asserts the existence of a universal machine that can simulate any instructions for a partial recursive function of lower arity. Intuitively, if we think that all the k-ary partial recursive functions are listed in one column, then all the information of that column, can be found in a single cell of in that entire numbering.

Theorem 1.2.6 (Enumeration theorem)**.** *There exists an e such that for all $y, x_1 \dots, x_k$,*

$$\phi_e(y,x_1,\dots,x_k) \simeq \phi_y(x_1,x_2,\dots,x_k)$$

.

Proof. The theorem follows immediately from theorem 1.2.5:

$$\psi(y, x_1, \dots, x_k) \equiv p(\mu z[t_k(y, x_1, \dots, x_k, z) = 1]) \simeq \phi_y(x_1, \dots, x_k).$$

By Church's Thesis, ψ has an index e. ■

Definition 1.2.7. A function f of k variables is *one-one* if, for every y, there is at most one k-tuple $\langle x_1, \dots, x_k \rangle$ such that $f(x_1, \dots, x_k) = y$.

The next result is also known as parametrization theorem and is in spirit the converse of the previous theorem.

Theorem 1.2.8 (*s-m-n* theorem)**.** *For every $m, n \geq 1$, there exists a one-one recursive function of $m + 1$ variables such that for all $e, y_1, \dots, y_m$,*

$$\lambda x_1 \dots x_n . \phi_e^{(m+n)}(y_1, \dots, y_m, x_1, \dots, x_n) \simeq \phi^{(n)}_{s_n^m(e, y_1, \dots, y_m)}. \tag{1.2.4}$$

Proof. Informally, let $P_{s_n^m(e,y_1,\dots,y_m)}$ on input $\overline{x}$ find P_e and run it on input $(\overline{y}, \overline{x})$. Hence, s_n^m is recursive by Church's Thesis. The function can be altered to a one-one function $\tilde{s}_n^m \equiv \lambda e y_1 \dots y_m . t(s_n^m(e, y_1, \dots, y_m), \langle e, y_1, \dots, y_m \rangle)$, where t is the recursive function defined in remark 1.2.4. A formal proof can be found in [8, §65]. ■

Corollary 1.2.9. *There exists a recursive function g of two variables such that for all x, y,*

$$\lambda z . \phi_{g(x,y)}(z) \simeq \lambda z . \phi_x \phi_y(z). \tag{1.2.5}$$

Proof. By Enumeration theorem there exists a universal machine $\psi(x, y)$ such that

$$\eta(x, y, z) \equiv \psi(x, \psi(y, z)) \simeq \phi_x(\phi_y(z)).$$

By Church's Thesis, the function η is recursive and has an index e say. It follows by *s-m-n* theorem that

$$\phi_{s_1^2(e,x,y)}(z) \simeq \eta_e(x, y, z) \simeq \phi_x(\phi_y(z)).$$

Now define $g(x, y) \equiv s_1^2(e, x, y)$. The result follows. ■

1.3 Undecidability of the halting problem

In this section we will show that there is no effective decision procedure that would give us a priori knowledge of whether an arbitrary machine on an arbitrary input would halt. That is, the halting problem is unsolvable.

Definition 1.3.1. A set A is called *recursive* or *decidable* if it possesses a recursive characteristic function,

$$\chi_A(x) \equiv \begin{cases} 1 & \text{if } x \in A; \\ 0 & \text{otherwise.} \end{cases} \tag{1.3.1}$$

Intuitively, a set A is recursive if we can uniform effectively decide its members.

Definition 1.3.2. A set A is called *recursively enumerable* (*r.e*) or *semidecidable* if it is the domain of a partial recursive function. Let the r.e set with Gödel number e be denoted by,

$$W_e \equiv \operatorname{dom} \phi_e = \{\, x \mid \phi_e(x)\downarrow \,\}. \tag{1.3.2}$$

By the Padding lemma, there are also infinitely many indices for a recursively enumerable set.

Remark 1.3.3 (Dovetailing)**.** We give a description for a useful construction that allows for simultaneous computation of possibly infinite inputs for a fixed p.r function ϕ_e, or for a fixed input x the simultaneous computation of possibly infinite p.r functions. In the first stage, perform one step of the computation of $\phi_e(0)$ (or $\phi_0(x)$), in the second stage perform the second step of $\phi_e(0)$ (or $\phi_0(x)$) and first step of $\phi_e(1)$ (or $\phi_1(x)$). Continue the pattern as in fig. 1.4, where $\phi_{e,s}(x)$ denotes s steps in the computation of $\phi_e(x)$ and $\phi_{e,0}(x)\uparrow$. These diagonal procedures are known as *dovetailing*.

$$\begin{tikzcd}
\phi_{e,1}(0) & \phi_{e,2}(0) \arrow[dl] & \phi_{e,3}(0) \arrow[dl] & \phi_{e,4}(0) \arrow[dl] & \cdots \\
\phi_{e,1}(1) & \phi_{e,2}(1) \arrow[dl] & \phi_{e,3}(1) \arrow[dl] & \cdots & \\
\phi_{e,1}(2) & \phi_{e,2}(2) \arrow[dl] & \ddots & & \\
\phi_{e,1}(3) & \vdots & & & \\
\vdots & & & &
\end{tikzcd}
\qquad
\begin{tikzcd}
\phi_{0,1}(x) & \phi_{0,2}(x) \arrow[dl] & \phi_{0,3}(x) \arrow[dl] & \phi_{0,4}(x) \arrow[dl] & \cdots \\
\phi_{1,1}(x) & \phi_{1,2}(x) \arrow[dl] & \phi_{1,3}(x) \arrow[dl] & \cdots & \\
\phi_{2,1}(x) & \phi_{2,2}(x) \arrow[dl] & \ddots & & \\
\phi_{3,1}(x) & \vdots & & & \\
\vdots & & & &
\end{tikzcd}$$

Figure 1.4: Illustration of dovetailing for inputs (left) and p.r functions (right).

Theorem 1.3.4 (Listing theorem)**.** *A set A is r.e if and only if $A = \emptyset$ or it is the range of a total recursive function η. Moreover, η can be found uniformly in an index e for a nonempty r.e set A.*

Proof. Let $A = \operatorname{dom} \phi_e$ and define a function ψ recursively by running the dovetailing procedure. Set,

$$\psi(e,0) \equiv \text{first } y \text{ such that } (\exists s)\phi_{e,s}(y)\downarrow;$$

$$\psi(e,x+1) \equiv \begin{cases} \text{first } y \text{ such that } (\exists y)_{y<x+1}[\phi_{e,x+1-y}(y)\downarrow \quad \wedge \quad (\forall w < x+1)[\phi_{e,x+1-y}(y) \neq \psi(e,w)]]; \\ \psi(e,0) \text{ otherwise,} \end{cases} \tag{1.3.3}$$

where $x+1$ is viewed as the diagonal stages in the dovetailing procedure. By Church's Thesis ψ is a partial recursive function. Indeed, whenever $A \neq \emptyset$, then $\eta \equiv \lambda x.\psi(e,x)$ is total and $A = \eta(\mathbb{N})$ uniformly in e. Note that each $x \in A$ is listed exactly once with exception for $\eta(0)$.

Conversely, if $A = \emptyset$ then it is the domain of the completely undefined partial recursive function $\lambda x.(\uparrow)$. If $A \neq \emptyset$, then $A = \eta(\mathbb{N})$ with η recursive. Now define $\psi \equiv \lambda x.\eta(\mu y[\eta(y) = x])$, which is partial recursive by Church's Thesis, then it is clear that $A = \operatorname{dom}\psi$. ■

It is in the sense of the Listing theorem that A is enumerable.

Theorem 1.3.5. *A set A is r.e if and only if it possesses a partial recursive characteristic function,*

$$\psi_A(x) \equiv \begin{cases} 1 & \text{if } x \in A; \\ \uparrow & \text{otherwise.} \end{cases} \tag{1.3.4}$$

Proof. Let $A = \operatorname{dom}\phi_e$ for some p.r function with index e. Then we have that

$$\psi_A(x) \equiv \begin{cases} 1 & \text{if } \phi_e(x)\downarrow; \\ \uparrow & \text{otherwise,} \end{cases}$$

is the desired p.r function by Church's Thesis. The converse is immediate: we have that $A = \operatorname{dom}\psi_A$, where ψ_A is p.r by definition. ■

It is in the above sense that A is semidecidable. Intuitively, our decision procedure is always limited to a partial answer as negative ones have a waiting time of forever.

Proposition 1.3.6. *If A is r.e and ψ is p.r, then $\psi^{-1}(A)$ and $\psi(A)$ are r.e.*

Proof. Suppose $A = \operatorname{dom}\phi_e$ for some e. It is clear that $\psi^{-1}(A) = \operatorname{dom}\phi_e\psi$. The result follows from corollary 1.2.9. We give a description of a partial recursive function whose domain is the image of A under ψ. For an input x, check if $\phi_e(x)$ converges and only then dovetail ψ on all inputs y and check if it converges and is equal to x, otherwise undefined. By Church's Thesis this defines a partial recursive partial whose domain is $\psi(A)$. ■

Lemma 1.3.7. *If A is recursive, then A is recursively enumerable.*

Proof. Let χ_A be the characteristic function of A. Let ψ be a partial recursive function defined by,

$$\psi(x) \equiv \begin{cases} 1 & \text{if } \chi_A(x) = 1; \\ \uparrow & \text{otherwise.} \end{cases}$$

Clearly, $A = \operatorname{dom}\psi$. ■

Theorem 1.3.8 (Post). *A set A is recursive if and only if A and $\overline{A}$ are recursively enumerable.*

Proof. It is immediate that if A is recursive, then its complement $\overline{A}$ is recursive; set $\chi_{\bar{A}}(x) \equiv 1 - \chi_A(x)$. The result follows by lemma 1.3.7.

Conversely, suppose $A = \operatorname{dom} \phi_e$ and $\overline{A} = \operatorname{dom} \phi_r$ for some e and r respectively. Define the recursive function

$$f(x) \equiv \mu s[\phi_{e,s+1}(x)\downarrow \vee \phi_{r,s}(x)\downarrow].$$

We are effectively dovetailing ϕ_e and ϕ_r. Now $x \in A$ if and only if $\phi_{e,f(x)}(x)\downarrow$, thus A is recursive. ■

Next we show that the converse of lemma 1.3.7 does not hold.

Definition 1.3.9. Let $K \equiv \{\, x \mid \phi_x(x)\downarrow \,\} = \{\, x \mid x \in W_x \,\}$ denote the *diagonal halting set.*

Theorem 1.3.10. *There exists a recursively enumerable but not recursive set, and K is such a set.*

Proof. Let $\phi_e(x, y)$ be the universal partial function of two variables from theorem 1.2.6. Then $K = \operatorname{dom} \psi$ with $\psi(x) \equiv \phi_e(x, x)$. Suppose towards a contradiction that K is recursive. Then we have that $\overline{K}$ is recursive, and therefore recursively enumerable by theorem 1.3.8. Let $\overline{K} = W_x$ for some x. It follows by definition of K and every choice of x that

$$x \in K \iff x \in W_x \iff x \in \overline{K} \iff x \notin K,$$

which is impossible. We conclude that K cannot be recursive. ■

While the class of recursive sets is closed under complementation, theorem 1.3.10 shows that the class of r.e sets is not. We are now in a position to take on the halting problem, which is encoded in the following way.

Definition 1.3.11. Let $K_0 \equiv \{\, \langle x,\ y\rangle \mid \phi_x(y)\downarrow \,\}$ denote the *halting set.*

Corollary 1.3.12. *The halting set K_0 is r.e, but not recursive.*

Proof. Indeed $K_0 = \operatorname{dom} \psi$ with $\psi(\langle x,\ y\rangle) \equiv \phi_e(x, y)$ in the Enumeration theorem 1.2.6. Note that $x \in K$ if and only if $\langle x,\ x\rangle \in K_0$. Thus an effective decision procedure for K_0 would imply the decidability of K, a contradiction. ■

The above proof gives an indirect technique for determining solvability of new problems by reducing a known unsolvable problem such as K to them. We can intuitively think of a problem being reducible to another if testing membership for the latter is not harder than the first.

Definition 1.3.13. Let A and B be sets.

(i) Write $A \leq_m B$ to mean A is *many-one reducible (m-reducible)* to B if there is a recursive function f such that $f(A) \subseteq B$ and $f(\overline{A}) \subseteq \overline{B}$;

(ii) and write $A \leq_1 B$ to mean A is *one-one reducible (1-reducible)* to B if there is a one-one recursive function f such that $f(A) \subseteq B$ and $f(\overline{A}) \subseteq \overline{B}$.

We write $\leq_r$ to refer to reducibility in general. Note that above condition is the same as stating $(\forall x)[x \in A \iff f(x) \in B]$ or $A = f^{-1}(B)$.

Proposition 1.3.14. *The following are basic properties of reducibility:*

(i) $\leq_r$ *is reflexive and transitive;*

(ii) if $A \leq_1 B$ *then* $A \leq_m B$*;*

(iii) if $A \leq_r B$ *then* $\overline{A} \leq_r \overline{B}$*;*

(iv) if $A \leq_r B$ *and* B *is recursive, then* A *is recursive;*

(v) if $A \leq_r B$ *and* B *is recursively enumerable, then* A *is recursively enumerable.*

Proof. In (i) transitivity follows from the fact that composition of recursive-and one-one functions is recursive- and one-one respectively. For reflexivity take the identity function $\lambda x.x$, which is recursive. Part (ii) is immediate by definition and (iii) follows from the fact that in **Set**, a set it equal to its double complement, so we can use the same f. Suppose χ_B is the recursive characteristic function of B in (iv), then $\chi_A = \chi_B f$. It follows by the proof of proposition 1.3.6 that $A = f^{-1}(B)$ in (v) is r.e. ■

Proposition 1.3.15. *Let* $Tot \equiv \{ x \mid \phi_x \text{ is total} \}$ *and* $K_1 \equiv \{ x \mid W_x \neq \emptyset \}$*, then* $K \leq_1 Tot$ *and* $K \leq_1 K_1$ *and are therefore not recursive.*

Proof. Define ψ by,

$$\psi(x,y) \equiv \begin{cases} 1 & \text{if } x \in K; \\ \uparrow & \text{otherwise.} \end{cases}$$

By Church's Thesis ψ is p.r with index z_0 say. Then by an application of *s-m-n* theorem 1.2.8 we have that $\psi(x,y) \simeq \phi_{f(x)}(y) \equiv \phi_{s^1_1(z_0,x)}$. Therefore,

$$\begin{aligned} x \in K &\implies \phi_{f(x)} \simeq \lambda x.(1) \implies f(x) \in Tot \quad (f(x) \in K_1); \text{ and} \\ x \in \overline{K} &\implies \phi_{f(x)} \simeq \lambda x.(\uparrow) \implies f(x) \in \overline{Tot} \quad (f(x) \in \overline{K_1}). \end{aligned}$$

Hence if Tot or K_1 were recursive then proposition 1.3.14 would render K recursive, which is impossible by theorem 1.3.10. ■

Remark 1.3.16. As K_1 is not decidable, the converse statement in theorem 1.3.4 cannot be proved uniformly effectively in an index for an r.e set A. In fact, these two notions do not coincide in $\mathcal{E}ff$.

It follows from (v) in proposition 1.3.14 that if $\overline{K} \leq_r A$ then A is not recursive or recursively enumerable. Not only is Tot undecidable, it is not even semidecidable.

Proposition 1.3.17. *We have that $\overline{K} \leq_r Tot$.*

Proof. We do this through a simple of dovetailing procedure. Define

$$\psi(x,s,y) \equiv \begin{cases} \uparrow & \text{if } \phi_{x,s}(x)\downarrow; \\ 1 & \text{if } \phi_{x,s}(x)\uparrow . \end{cases}$$

It has an index z_0 say, by Church's Thesis. Through an application of *s-m-n* define $f(x,s) \equiv s^2_1(z_0, x, s)$. Then $g \equiv \lambda z.f(\pi_1(z), \pi_2(z))$, is a recursive one-one function of one variable by proposition 1.1.11, where $\psi(x,s,y) \simeq \phi_{g(\langle x,\, s\rangle)}(y)$. Then it is apparent that,

$$z \in \overline{K} \implies \phi_{g(z)} = \lambda x.(1) \implies g(z) \in Tot; \text{ and}$$

$$z \in K \implies \phi_{g(z)} = \lambda x.(\uparrow) \implies g(z) \in \overline{Tot}. \qquad \blacksquare$$

We next show two important closure properties of recursively enumerable set. Recursively enumerable sets are closed under infinite union and finite intersection indexed by a recursively enumerable set. Note that finite sets are r.e: we can just give a finite description of a partial recursive function with that domain.

Proposition 1.3.18. *Let I be a possibly infinite recursively enumerable set and F a finite set, then $\bigcup\limits_{n\in I} W_n$ and $\bigcap\limits_{n\in F} W_n$ are recursively enumerable.*

Proof. For a fixed x run a dovetailing procedure on partial recursive functions ϕ_n with $n \in A$. The process will terminate if and only if there an n such that $\phi_{n,s}(x)\downarrow$ in some steps s. Thus $\bigcup\limits_{n\in I} W_n$ is the domain of such a partial recursive function by Church's Thesis. Similarly, dovetail the finite collection of partial recursive functions ϕ_n with $n \in F$, the process will terminate if and only if every $\phi_{n,s}(x)$ converges for some s. Thus $\bigcap\limits_{n\in F} W_n$ is the domain of such a partial recursive function by Church's Thesis. $\blacksquare$

Remark 1.3.19. Intersection of r.e sets under an r.e index is not necessarily r.e. Intuitively, we can think of it as the dovetailing procedure to be in vain as possibly an infinite number of steps s will still have to be checked. So, even if there is an answer or the intersection is nonempty, we would have to wait forever. We give an explicit construction. The set $\{ x \mid \phi_x(n)\downarrow \}$ is r.e. as it is the domain of $\psi(x) \equiv \phi_e(x,n)$ with ϕ_e the universal machine described in theorem 1.2.6. Yet,

$$\bigcap_{n\in\mathbb{N}} \{ x \mid \phi_x(n)\downarrow \} = Tot, \tag{1.3.5}$$

which is not r.e by proposition 1.3.17. In fact, we have shown failure of closure under infinite intersection indexed by a recursive set.

K is in a rather strong sense nonrecursive. Namely, that it in a sense has the highest degree of unsolvability among the r.e sets.

Definition 1.3.20. A set A is *r-complete* with respect to $\leq_r$ if

(i) A is recursively enumerable;

(ii) $(\forall B)[B \text{ recursively enumerable} \implies B \leq_r A]$.

Proposition 1.3.21. *The sets K, K_0 and K_1 are r-complete.*

Proof. Let B be any recursively enumerable set, then $B = W_e$ for some e. It is immediate that $x \in W_e$ if and only if $\langle e, x\rangle \in K_0$, thus K_0 is complete. It suffices to show that $K_0 \leq_r K$ by transitivity; define

$$\psi(x,y) \equiv \begin{cases} 1 & \text{if } \phi_{\pi_1(x)}(\pi_2(x))\downarrow; \\ \uparrow & \text{otherwise,} \end{cases}$$

then ψ has an index z_0 by Church's Thesis. A tacit application of *s-m-n* theorem yields that $\psi(x,y) \simeq \phi_{f(x)}(y)$. We have,

$$\begin{aligned} x \in K_0 &\implies \phi_{f(x)} = \lambda x.(1) \implies \phi_{f(x)}(f(x))\downarrow \implies f(x) \in K; \text{ and} \\ x \in \overline{K_0} &\implies \phi_{f(x)} = \lambda x.(\uparrow) \implies \phi_{f(x)}(f(x))\uparrow \implies f(x) \in \overline{K}. \end{aligned}$$

Again, K_1 is r-complete by transitivity and proposition 1.3.15. ■

Definition 1.3.22 (Dekker)**.** A set P is said to be *productive* if it possesses a productive partial recursive function ψ such that

$$(\forall x)[W_x \subseteq P \implies [\psi(x)\downarrow \quad \wedge \quad \psi(x) \in P - W_x]].$$

The set $\overline{K}$, on the other hand, is not recursively enumerable in a very strong sense as well. We are saying that P is not recursive and moreover there is a uniformly effective proof of this fact in: for every x, the function $\psi(x)$ gives a counterexample.

Proposition 1.3.23. *The set $\overline{K}$ is productive.*

Proof. Let $\psi \equiv \lambda x.x$. Note that the identity function is recursive. Then given an x, if $W_x \subseteq \overline{K}$ then $\phi_x(x)\uparrow$ and thus $x \in \overline{K} - W_x$ trivially. ■

Proposition 1.3.24. *The following are basic properties of productive sets:*

(i) if P is productive then P is not recursively enumerable;

(ii) if P is productive and $P \leq_m A$, then A is productive.

Proof. Part (i) is immediate by definition. Let ψ denote the product p.r function for P and let $P \leq_m A$ via f with fixed index z_0. Suppose $W_x \subseteq A$, then $W_{h(x)} = f^{-1}(W_x) \subseteq f^{-1}(A) = P$ by productiveness and by proposition 1.3.6, where $h(x) = g(x, z_0)$ from corollary 1.2.9. It follows that $\psi h(x)\downarrow$ and $\psi h(x) \in P - W_{h(x)}$. Thus f is witness that $f\psi h(x) \in A$ and $f\psi h(x) \notin W_x$ by $W_{h(x)} = f^{-1}(W_x)$. Hence $f\psi h$ defines a productive p.r function for A. ■

Thus Tot is productive.

Proposition 1.3.25. *A set P is productive if and only if P has a recursive productive function p.*

Proof. Let ψ be a partial productive function of P. Define

$$\eta(x, y) \equiv \begin{cases} 1 & \text{if } \psi(x)\downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

By a usual application of *s-m-n* and Church's Thesis, we have that $\eta(x, y) \simeq \phi_{g(x)}(y)$. Taking domains we get,

$$W_{g(x)} = \begin{cases} W_x & \text{if } \psi(x)\downarrow; \\ \emptyset & \text{otherwise.} \end{cases}$$

To define p, dovetail ψ on inputs $g(x)$ and x and take the output to be the first to converge. The function p is total as $\emptyset \subseteq P$, thus there will always be an output.

The converse is immediate by definition. ■

Definition 1.3.26 (Post)**.** A set C is said to be *creative* if

(i) C is recursively enumerable;

(ii) $\overline{C}$ is productive.

Thus K is creative. There is an equivalent characterisation.

Proposition 1.3.27 (Myhill)**.** *A set C is creative if and only if C is recursively enumerable and there exists a unary partial recursive function u such that*

$$(\forall x)[W_x \cap C = \emptyset \implies [u(x)\downarrow \quad \wedge \quad u(x) \notin W_x \cup C]]. \tag{1.3.6}$$

Proof. Immediate by writing out the definitions and using the same p.r function. ■

Proposition 1.3.28. *The following are properties of creative sets:*

(i) if C is creative then C is not recursive;

(ii) if C is creative and $C \leq_m A$, then $\overline{A}$ is productive;

(iii) if A is m-complete, then A is creative.

Proof. Part (i) follows from theorem 1.3.8 and the fact that $\overline{C}$ is productive and thus not r.e by definition. Recall from proposition 1.3.14 that $C \leq_m A$ implies that $\overline{C} \leq_m \overline{A}$, and thus (ii) follows from (ii) of proposition 1.3.24. For (iii) it suffices to show that $\overline{A}$ is productive as A is r.e by definition. But A is m-complete, so in particular $K \leq_m A$. The result follows by part (ii). ■

The content of Myhill's theorem is the converse of (iii) in proposition 1.3.28. Before we can prove that creative sets are complete, we need to establish a few results that are interesting in their own right.

Theorem 1.3.29 (Fixed point theorem). *For every recursive function f there exists an n such that*

$$\phi_{f(n)}(x) \simeq \phi_n(x).$$

Such an n is called a fixed point of f.

Proof. Define a partial recursive function ψ by,

$$\psi(u,x) \equiv \begin{cases} \phi_{\phi_u(u)}(x) & \text{if } \phi_u(u)\downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

By *s-m-n* theorem 1.2.8 we have that $\psi(u,v) \simeq \phi_{g(u)}(x) \equiv \phi_{s^1_1(z_0,u)}(x)$, where z_0 is an Gödel number for ψ by Church's Thesis. Then fg is recursive with an index v say. As $\phi_v \simeq fg$ is total, $\phi_v(v)\downarrow$ so that

$$\phi_{fg(v)}(x) \equiv \phi_{\phi_v(v)}(x) \simeq \phi_{g(v)}.$$

Thus take $n \equiv g(v)$ as the desired fixed point. ■

The proof of the first recursion theorem is quite odd. It looks like a diagonal argument, but it fails. Imagine an infinite matrix with rows $R_i = \{ \phi_{\phi_i(u)} \}_{u\in\mathbb{N}}$. The diagonal is precisely $D = \{ \phi_{\phi_u(u)} \}_{u\in\mathbb{N}}$ with $\phi_{\phi_u(u)} = \lambda x.(\uparrow)$ whenever $\phi_u(u)\uparrow$. Yet we fail to generate any indices beyond those that already exist as

$$D = \{ \phi_{\phi_u(u)} \}_{u\in\mathbb{N}} = \{ \phi_{g(u)} \}_{u\in\mathbb{N}} = \{ \phi_{\phi_e(u)} \}_{u\in\mathbb{N}} = R_e.$$

This is due to the strong properties of S^m_n that recovers the data. Now it is clear that $R_v = \{ \phi_{\phi_v(u)} \}_{u\in\mathbb{N}} = \{ \phi_{fg(u)} \}_{u\in\mathbb{N}}$ and D must coincide on (v,v).

Corollary 1.3.30. *For every recursive function f there exists an n such that*

$$W_{f(n)} = W_n.$$

Proof. Immediate by the the first recursion theorem 1.3.29. ■

Corollary 1.3.31. *There exists an n such that*

$$W_n = \{ n \}.$$

Proof. Define a function ψ by,

$$\psi(x, y) \equiv \begin{cases} 1 & \text{if } x = y; \\ \uparrow & \text{otherwise.} \end{cases}$$

By Church's Thesis and the usual application of *s-m-n* we have that $\psi(x, y) \simeq \phi_{f(x)}(y)$ so that for all x, $W_{f(x)} = \{ x \}$. Then theorem 1.3.29 asserts the existence of an n such that $W_n = W_{f(n)} = \{ n \}$. ■

In fact the first recursion theorem has the property that we can find n uniformly in a Gödel number for f whenever f is total.

Proposition 1.3.32. *There exists a recursive unary function n such that for any z, if ϕ_z is total, then*

$$\phi_{\phi_z(n(z))} \simeq \phi_{n(z)}.$$

Proof. Let $f \equiv \phi_z$. By corollary 1.2.9 we can obtain a Gödel number for fg in the proof of theorem 1.3.29 uniformly from z. Let $\phi_{v(z)} \simeq fg$, then define $n(z) \equiv gv(z)$ as the desired recursive function. ■

Actually by padding the indices, viz. defining $\tilde{v}(z) = t(v(z), z)$, with t in remark 1.2.4 we can get n to be one-one. In order to prove Myhill's theorem, we need to establish a stronger version of the Fixed point theorem, which is parametrised.

Theorem 1.3.33 (Kleene's second recursion theorem)**.** *For each k, there exists a recursive function n of $k + 1$ variables such that for any z, if $\phi_z^{(k+1)}$ is total, then for all $x_1, \dots, x_k$*

$$\phi_{\phi_z^{(k+1)}(n(z,x_1,\dots,x_k),x_1,\dots,x_k)} \simeq \phi_{n(z,x_1,\dots,x_k)}.$$

Proof. Define a partial recursive function ψ by,

$$\psi(u, \overline{x}, \overline{y}) \equiv \begin{cases} \phi_{\phi_u(u,\overline{x})} & \text{if } \phi_u(u, \overline{x})\downarrow; \\ \uparrow & \text{otherwise.} \end{cases}$$

By *s-m-n* theorem 1.2.8 we have that $\psi(u, \overline{x}, \overline{y}) \simeq \phi_{g(u,\overline{x})} \equiv \phi_{s_n^{(k+1)}(z_0,u,\overline{x})}$, where z_0 is a Gödel number for ψ by Church's Thesis. Let v be a recursive function such that $v(z)$ is a Gödel number for the total recursive function $\phi_z(g(u, \overline{x}), \overline{x})$, then $\phi_{v(z)}(v(z), \overline{x})\downarrow$. Now define $n(z, \overline{x}) = g(v(z), \overline{x})$. This is the desired fixed point, for

$$\phi_{\phi_z(g(v(z),\overline{x}),\overline{x})} \simeq \phi_{\phi_{v(z)}(v(z),\overline{x})} \simeq \phi_{g(v(z),\overline{x})}.$$ ■

Theorem 1.3.34 (Myhill). *A set C is creative if and only if C is complete.*

Proof. We will show that $\overline{K}$ is a kind of lower bound for productiveness, viz. $\overline{K} \leq_m P$ for any productive set P. Fix a P, then by proposition 1.3.25 it possesses a productive recursive function p. Define

$$\psi(x, y, z) \equiv \begin{cases} 1 & \text{if } \phi_y(y)\downarrow \quad \wedge \quad p(x) = z; \\ \uparrow & \text{otherwise.} \end{cases}$$

A tacit application of Church's thesis and *s-m-n* yields $\psi(x, y, z) \simeq \phi_{f(x,y)}(z)$. Taking domains we get,

$$W_{f(x,y)} = \begin{cases} \{ p(x) \} & \text{if } y \in K; \\ \emptyset & \text{otherwise.} \end{cases}$$

Theorem 1.3.33 asserts the existence of a recursive function n such that

$$W_{n(x)} = W_{f(n(x),x)} = \begin{cases} \{ p(n(x)) \} & \text{if } x \in K; \\ \emptyset & \text{otherwise.} \end{cases}$$

It follows that

$x \in \overline{K} \implies W_{n(x)} = \emptyset \implies p(n(x)) \in P$; and
$x \in K \implies W_{n(x)} = \{ p(n(x)) \} \implies p(n(x)) \notin P - W_{n(x)} \implies W_{n(x)} \nsubseteq P \implies p(n(x)) \in \overline{P}$.

Thus C is creative implies $\overline{K} \leq_m \overline{C}$. The desired result follows from (ii) and (iii) of proposition 1.3.14, and proposition 1.3.21. ∎

Chapter 2

Crossing Over to the Effective Topos

2.1 Intuitionistic logic in the language of categories

Intuitionistic logic is a logic of constructive mathematics, where a proposition is not taken to have an intrinsic truth-value unless an explicit proof of it has been constructed and an object exists only when its proof also exhibits a way to find it. The former leads to the rejection of the law of excluded middle as a principle that should universally hold for every proposition. On these grounds, proofs by contradiction must also be rejected as we shall see. We begin by studying Lambek's axiomatisation of intuitionistic logic in which proof theory is put in the syntactical framework of objects and arrows following the book by Lambek and Scott [11]. A more standard treatment can be found in the following book by van Dalen [3]. We derive the usual rules of propositional calculus, which hold intuitionistically.

Definition 2.1.1. A *graph* $\mathcal{G}$ is the data (X, Y, s, t) consisting of a set of *arrows* (oriented edges) X and a set of *objects* (nodes) Y and a pair of functions $s, t\colon X \to Y$ assigning to each arrow f its *source* $s(f) = A$ and *target* $t(f) = B$, denoted by $A \xrightarrow{f} B$ or $f\colon A \to B$.

To a logician a directed multigraph with loops, in which reflexivity and transitivity of entailment is captured, is of particular interest. We call such structures deductive systems.

Definition 2.1.2. A *deductive system* $\mathcal{D}$ is a triple $(\mathcal{G}, 1, \circ)$ consisting of a graph $\mathcal{G}$ such that for each object A there exists an arrow $1_A\colon A \to A$, and for each pair of arrows $f\colon A \to B$ and $g\colon B \to C$ there is an arrow $g \circ f\colon A \to C$.

We view the objects of the deductive system as *formulas* and arrows as *proofs* or *deductions*. In the proof $f\colon A \to B$, we call A the *antecedent*, B the *consequent*, and f the *witness* of entailment. Operations on arrows are seen as *inference rules* and specified arrows as *axiom schema*. We will follow the custom of denoting composition $g \circ f$ by juxtaposition gf.

By further imposing additional structure with binary operations *and* $\wedge$, *or* $\vee$ and *implies* $\Rightarrow$, and specified objects *true* $\top$ and *false* $\perp$ we can capture the full intuitionistic propositional calculus. Thus, as in the definition for a deductive system we insist that for each object certain arrows exist, viz. for each operation there are *introduction rules* **I**, which show how formulas can be derived and *elimination rules* **E**, which show how formulas can be used to derive others. We summarise the above in the following definition.

Definition 2.1.3. The following are axiom schema and rules of inference for intuitionistic propositional logic:

R1. $$\frac{}{A \xrightarrow{1_A} A} \quad \textbf{(Taut)}$$

R2. $$\frac{A \xrightarrow{f} B \qquad B \xrightarrow{g} C}{A \xrightarrow{gf} C} \quad \textbf{(Cut)}$$

R3. $$\frac{}{A \xrightarrow{\bigcirc_A} \top} \quad \textbf{(T)}$$

R4a. $$\frac{}{A \wedge B \xrightarrow{\pi_{A,B}} A} \quad (\wedge\textbf{E}_1)$$

R4b. $$\frac{}{A \wedge B \xrightarrow{\pi'_{A,B}} B} \quad (\wedge\textbf{E}_2)$$

R4c. $$\frac{C \xrightarrow{f} A \qquad C \xrightarrow{g} B}{C \xrightarrow{\langle f,g\rangle} A \wedge B} \quad (\wedge\textbf{I})$$

R5a. $$\frac{}{(B \Rightarrow A) \wedge B \xrightarrow{\varepsilon_{A,B}} A} \quad (\Rightarrow\textbf{E})$$

R5b. $$\frac{A \wedge B \xrightarrow{f} C}{A \xrightarrow{\overline{f}} B \Rightarrow C} \quad (\Rightarrow\textbf{I})$$

R6. $$\frac{}{\perp \xrightarrow{\square_A} A} \quad \textbf{(EFQ)}$$

R7a. $$\frac{}{A \xrightarrow{\kappa_{A,B}} A \vee B} \quad (\vee\textbf{I}_1)$$

R7b. $$\frac{}{B \xrightarrow{\kappa'_{A,B}} A \vee B} \quad (\vee\textbf{I}_2)$$

R7c. $$\frac{}{(A \Rightarrow C) \wedge (B \Rightarrow C) \xrightarrow{\xi^C_{A,B}} (A \vee B) \Rightarrow C} \quad (\vee\textbf{E})$$

Axiom schema of classical propositional logic:

R8. $$\frac{}{(A \Rightarrow \bot) \Rightarrow \bot \xrightarrow{\nu_A} A} \quad \textbf{(DNE)}$$

The rules R1.–R4 and R1.–R5. constitute a conjunction- and a positive intuitionistic calclulus respectively.

Composing operations on arrows give rise to *derived rules.* We say that an inference rule is *admissable* whenever there are proofs of the premise, there is a proof of the conclusion. We can therefore admit these rules without changing the specified rules of the calclulus.

Lemma 2.1.4. *The following derived rules of inference are admissable.*

$$\frac{A \xrightarrow{f} B \qquad C \xrightarrow{g} D}{A \wedge C \xrightarrow{f \wedge g} B \wedge D} \tag{2.1.1}$$

$$\frac{B \xrightarrow{f} C}{A \Rightarrow B \xrightarrow{1_A \Rightarrow f} A \Rightarrow C} \tag{2.1.2}$$

$$\frac{A \xrightarrow{f} B \qquad C \xrightarrow{g} D}{D \Rightarrow A \xrightarrow{g \Rightarrow f} C \Rightarrow B} \tag{2.1.3}$$

$$\frac{A \xrightarrow{f} B}{\top \xrightarrow{\ulcorner f \urcorner} A \Rightarrow B} \tag{2.1.4}$$

$$\frac{\top \xrightarrow{f} A \Rightarrow B}{A \xrightarrow{f^{\wr}} B} \tag{2.1.5}$$

$$\frac{A \xrightarrow{f} C \qquad B \xrightarrow{g} C}{A \vee B \xrightarrow{[f,g]} C} \tag{2.1.6}$$

$$\frac{A \xrightarrow{f} B \qquad C \xrightarrow{g} D}{A \vee C \xrightarrow{f \vee g} B \vee D} \tag{2.1.7}$$

Proof. We give an illustration of the first derived rule.

$$\frac{\dfrac{(\wedge\mathrm{E}_1)\ A \wedge C \xrightarrow{\pi_{A,B}} A \qquad A \xrightarrow{f} B}{A \wedge C \xrightarrow{f\pi_{A,B}} B}\,\textbf{(Cut)} \qquad \dfrac{(\wedge\mathrm{E}_2)\ A \wedge C \xrightarrow{\pi'_{A,B}} C \qquad C \xrightarrow{g} D}{A \wedge C \xrightarrow{g\pi'_{A,B}} D}\,\textbf{(Cut)}}{A \wedge C \xrightarrow{\langle f\pi_{A,B},\, g\pi'_{A,B}\rangle} B \wedge D}\,(\wedge\mathrm{I})$$

Let $f \wedge g \equiv \langle f\pi_{A,B}, g\pi'_{A,B}\rangle$, now we can abbreviate the above derivation to the inference rule (2.1.1). We will only define the remaining rules. It is straight forward to retrieve the derivation from the definition. Next set $1_A \Rightarrow f \equiv \overline{f\varepsilon_{B,A}}$, from which it follows that $g \Rightarrow f \equiv \overline{\varepsilon_{B,D}((1_D \Rightarrow f) \wedge g)}$. Define *name of* f as $\ulcorner f \urcorner \equiv \overline{f\pi_{\top\wedge A}}$ and *application* as $f^{\wr} \equiv \varepsilon_{B,A}\langle f\bigcirc_A, 1_A\rangle$. The last two derived rules are thus given by $[f,g] \equiv (\xi^C_{A,B}\langle \ulcorner f \urcorner, \ulcorner g \urcorner\rangle)^{\wr}$ and $f \vee g \equiv [\kappa_{B,D} f,\ \kappa'_{B,D} g]$. ■

Note how implication is tied to adjunction in definition 2.1.3 as opposed to negation and disjunction in the classical case. For instance, modus ponens (⇒**E**) resembles the counit of adjunction. Indeed, we can derive what resembles the unit of adjunction,

$$\frac{}{A \xrightarrow{\eta_{A,B}} B \Rightarrow (A \wedge B)} \tag{2.1.8}$$

where $\eta_{A,B} \equiv \overline{1_{A \wedge B}}$. Conversely, given (2.1.8) and (2.1.2) we can replace the transpose (rule R5b.) in definition 2.1.3 by letting $\overline{f} \equiv (1_A \Rightarrow f)\eta_{A,B}$. Next, we derive a few familiar rules of propositional calculus.

Lemma 2.1.5. *The following proofs are valid in intuitionistic propositional calculus for each object. The subsequent arrows are pairwise converse.*

$$\frac{}{A \wedge \top \xrightarrow{\pi_{A,\top}} A} \tag{2.1.9}$$

$$\frac{}{A \xrightarrow{\langle 1_A,\, \circ_A \rangle} A \wedge \top} \tag{2.1.10}$$

$$\frac{}{\top \Rightarrow A \xrightarrow{\varepsilon_{A,\top}\langle 1_{\top \Rightarrow A},\, \circ_{\top \Rightarrow A} \rangle} A} \tag{2.1.11}$$

$$\frac{}{A \xrightarrow{\overline{\pi_{A,\top}}} \top \Rightarrow A} \tag{2.1.12}$$

$$\frac{}{A \Rightarrow \top \xrightarrow{\overline{\circ_{A \Rightarrow \top}}} \top} \tag{2.1.13}$$

$$\frac{}{\top \xrightarrow{\overline{\pi_{\top,A}}} A \Rightarrow \top} \tag{2.1.14}$$

$$\frac{}{A \wedge \bot \xrightarrow{\pi'_{A,\bot}} \bot} \tag{2.1.15}$$

$$\frac{}{\bot \xrightarrow{\langle \square_A,\, 1_\bot \rangle} A \wedge \bot} \tag{2.1.16}$$

$$\frac{}{\bot \Rightarrow A \xrightarrow{\circ_{\bot \Rightarrow A}} \top} \tag{2.1.17}$$

$$\frac{}{\top \xrightarrow{\ulcorner \square_A \urcorner} \bot \Rightarrow A} \tag{2.1.18}$$

$$\frac{}{A \xrightarrow{\kappa_{A,\bot}} A \vee \bot} \tag{2.1.19}$$

$$\frac{}{A \vee \bot \xrightarrow{[1_A,\, \square_A]} A} \tag{2.1.20}$$

It is possible to derive absurdity from contradiction,

$$\frac{}{A \wedge (A \Rightarrow \bot) \xrightarrow{\varepsilon_{\bot,A}\langle \pi'_{A,A\Rightarrow\bot},\, \pi_{A,A\Rightarrow\bot}\rangle} \bot} \tag{2.1.21}$$

and double negation introduction (**DNI**) *is a valid rule.*

$$\frac{}{A \xrightarrow{\overline{\varepsilon_{\bot,A}\langle \pi'_{A,A\Rightarrow\bot},\, \pi_{A,A\Rightarrow\bot}\rangle}} (A \Rightarrow \bot) \Rightarrow \bot} \tag{2.1.22}$$

Proof. The proof is in the pudding. ∎

Lemma 2.1.6. *Distributivity of conjunction over implication is a valid rule of intuitionistic propositional calculus.*

$$\frac{}{C \Rightarrow (A \wedge B) \rightarrow (C \Rightarrow A) \wedge (C \Rightarrow B)} \tag{2.1.23}$$

$$\frac{}{(C \Rightarrow A) \wedge (C \Rightarrow B) \rightarrow C \Rightarrow (A \wedge B)} \tag{2.1.24}$$

Proof. The first derivation is given by $\langle \overline{\pi_{A,B}\varepsilon_{A\wedge B,C}},\, \overline{\pi'_{A,B}\varepsilon_{A\wedge B,C}} \rangle$ and the derivation of the second arrow is given by $\overline{\langle \varepsilon_{A,C}(\pi_{(C\Rightarrow A)\wedge(C\Rightarrow B)} \wedge 1_C),\, \varepsilon_{B,C}(\pi'_{(C\Rightarrow A)\wedge(C\Rightarrow B)} \wedge 1_C)\rangle}$. ∎

Lemma 2.1.7. *Commutativity and associativity of conjunction is a valid rule of intuitionistic propositional calculus.*

$$\frac{}{(A \wedge B) \rightarrow B \wedge A} \tag{2.1.25}$$

$$\frac{}{(A \wedge B) \wedge C \xrightarrow{\alpha_{A,B,C}} A \wedge (B \wedge C)} \tag{2.1.26}$$

$$\frac{}{A \wedge (B \wedge C) \xrightarrow{\alpha'_{A,B,C}} (A \wedge B) \wedge C} \tag{2.1.27}$$

Proof. Commutativity is given by $\langle \pi'_{A,B},\, \pi_{A,B} \rangle$. Associativity is given by,

$$\alpha_{A,B,C} \equiv \langle \pi_{A,B}\pi_{A\wedge B,C},\, \langle \pi'_{A,B}\pi_{A\wedge B,C},\, \pi'_{A\wedge B,C} \rangle\rangle; \text{ and symmetrically}$$

$$\alpha'_{A,B,C} \equiv \langle\langle \pi_{A,B\wedge C},\, \pi_{B,C}\pi'_{A,B\wedge C}\rangle,\, \pi'_{B,C}\pi'_{A,B\wedge C}\rangle.$$

∎

Lemma 2.1.8. *Exportation is a valid rule of intuitionistic propositional calculus.*

$$\frac{}{(A \wedge B) \Rightarrow C \rightarrow A \Rightarrow (B \Rightarrow C)} \tag{2.1.28}$$

$$\frac{}{A \Rightarrow (B \Rightarrow C) \rightarrow (A \wedge B) \Rightarrow C} \tag{2.1.29}$$

Proof. Let α be defined as in lemma 2.1.7, then the first proof is given by $\overline{\overline{\varepsilon_{C,A\wedge B}\alpha_{(A\wedge B)\Rightarrow C,A,B}}}$, and the second follows by $\overline{\varepsilon_{C,B}(\varepsilon_{B\Rightarrow C,A}\wedge 1_B)\alpha'_{A\Rightarrow(B\Rightarrow C),A,B}}$. ■

Lemma 2.1.9. *Conjunction distributes over disjunction in intuitionistic propositional calculus.*

$$\frac{}{(A\wedge C)\vee(B\wedge C)\xrightarrow{\delta_{A,B,C}}(A\vee B)\wedge C}\tag{2.1.30}$$

$$\frac{}{(A\vee B)\wedge C\xrightarrow{\delta'_{A,B,C}}(A\wedge C)\vee(B\wedge C)}\tag{2.1.31}$$

Proof. Define $\delta_{A,B,C}\equiv[\kappa_{A,B}\wedge 1_C,\ \kappa'_{A,B}\wedge 1_C]$ and set $\delta'_{A,B,C}\equiv\varepsilon_{(A\wedge C)\vee(B\wedge C),C}[\overline{\kappa_{A\wedge C,B\wedge C}},\ \overline{\kappa'_{A\wedge C,B\wedge C}}]$. ■

For the sake of brevity we introduce a new notation for negation.

Definition 2.1.10. For each formula A, define $\neg A\equiv A\Rightarrow\bot$, read *not* A.

Lemma 2.1.11. *The following two rules are derivable in intuitionistic proposition calculus:*

$$\frac{}{\neg A\to\neg\neg\neg A}\tag{2.1.32}$$

$$\frac{}{\neg\neg\neg A\to\neg A}\tag{2.1.33}$$

Proof. For the first proof replace the instance A by $\neg A$ in (2.1.22). The following derivation yields the converse.

$$\dfrac{\dfrac{\text{(Taut)}\;\neg\neg\neg A\to\neg\neg\neg A\qquad\text{(DNI)}\;A\to\neg\neg A}{\neg\neg\neg A\wedge A\to\neg\neg\neg A\wedge\neg\neg A}\,\text{(2.1.1)}\qquad(\Rightarrow\text{E})\;\neg\neg\neg A\wedge\neg\neg A\to\bot}{\dfrac{\neg\neg\neg A\wedge A\to\bot}{\neg\neg\neg A\to\neg A}\,(\Rightarrow\text{I})}\,\text{(Cut)}$$

■

Lemma 2.1.12. *The converse of* ($\vee$**I**) *in definition 2.1.3 is a valid rule of intuitionistic propositional calculus,*

$$\frac{}{(A\vee B)\Rightarrow C\to(A\Rightarrow C)\wedge(B\Rightarrow C)}\tag{2.1.34}$$

and thus De Morgan's law hold.

$$\frac{}{\neg(A\vee B)\to\neg A\wedge\neg B}\tag{2.1.35}$$

$$\frac{}{\neg A\wedge\neg B\to\neg(A\vee B)}\tag{2.1.36}$$

Proof. The converse is given by $\langle\kappa_{A,B}\Rightarrow 1_C,\ \kappa'_{A,B}\Rightarrow 1_C\rangle$. Now replace the instance of C by $\bot$ in ($\vee$**I**) in definition 2.1.3 and similarly in (2.1.34) to obtain De Morgan's law. ■

Lemma 2.1.13. *Disjunctive syllogism is a valid rule of intuitionistic propositional calculus.*

$$\dfrac{}{(A\vee B)\wedge(B\Rightarrow\bot)\to A} \tag{2.1.37}$$

Proof. Let δ be the arrow defined as in lemma 2.1.9, then the derivation of the above arrow is as follows $[1_A,\ \square_A](\pi_{A,B\Rightarrow\bot}\vee(\varepsilon_{\bot,B}\langle\pi'_{B,B\Rightarrow\bot},\ \pi_{B,B\Rightarrow\bot}\rangle))\delta_{A,B,B\Rightarrow\bot}$. ■

Proposition 2.1.14. *In the presence of rules R1 – R8 , the classical axiom* (**DNE**) *can be replaced by the law of excluded middle* (**LEM**) $\top\to A\vee(A\Rightarrow\bot)$ *for each object A.*

Proof. The following derivation shows that LEM follows from DNE. We split the derivation into two parts for the sake of presentation.

$$\dfrac{\dfrac{(2.1.35)\ \neg(A\vee\neg A)\to\neg A\wedge\neg\neg A \qquad (2.1.25)\ \neg A\wedge\neg\neg A\to\neg\neg A\wedge\neg A}{\neg(A\vee\neg A)\to\neg\neg A\wedge\neg A}\,(\mathbf{Cut}) \qquad (\Rightarrow\mathbf{E})\ \neg\neg A\wedge\neg A\to\bot}{\neg(A\vee\neg A)\to\bot}\,(\mathbf{Cut})$$
$$\vdots$$

$$\dfrac{\dfrac{\begin{matrix}\vdots\\ \neg(A\vee\neg A)\to\bot\end{matrix}}{\top\to\neg\neg(A\vee\neg A)}\,(2.1.4) \qquad (\mathbf{DNE})\ \neg\neg(A\vee\neg A)\to A\vee\neg A}{\top\to A\vee\neg A}\,(\mathbf{Cut})$$

Conversely, we have that DNE follows from LEM by the following derivation.

$$\dfrac{\dfrac{\dfrac{(\mathbf{LEM})\ \top\to A\vee\neg A \qquad (\mathbf{Taut})\ \neg\neg A\to\neg\neg A}{\top\wedge\neg\neg A\to(A\vee\neg A)\wedge\neg\neg A}\,(2.1.1) \qquad (2.1.37)\ (A\vee\neg A)\wedge\neg\neg A\to A}{\top\to\neg\neg A\Rightarrow A}\,(\mathbf{Cut})}{\neg\neg A\to A}\,(2.1.5)$$

■

In the previous chapter, we used the term *uniformly effectively* to mean different things. For instance in remark 1.3.16 it meant that the disjunction was undecidable. Intuitively, truth is taken to carry some data and some amount of 'work' must be done to establish existence. The Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic makes this precise [3, 5.1].

Definition 2.1.15 (BHK interpretation)**.** Let φ be a sentence of Heyting arithmetic. We say that e *proves* φ if e verifies φ by an explicit construction. Define inductively,

($\wedge$) e proves $\varphi\wedge\psi$ if and only if e is a pair $\langle n,\ m\rangle$ such that n proves ψ and m proves ψ;

($\vee$) e proves $\varphi\vee\psi$ if and only if e is a pair $\langle n,\ m\rangle$ with $n\in\mathbb{N}$ such that either $n=0$ and m proves φ or $n=1$ and m proves ψ;

($\to$) e proves $\varphi\to\psi$ if and only if e is a construction that converts every proof n of φ into a proof $e(n)$ of ψ;

($\bot$) no e proves $\bot$;

and introduce quantifiers over a domain D of objects for which,

($\forall$) e proves $\forall x.\varphi(x)$ if and only if e is a construction such that for each $n \in D$, $e(n)$ proves $\varphi(\underline{n})$;

($\exists$) e proves $\exists x.\varphi(x)$ if and only if e is a pair $\langle n, m\rangle$ such that $n \in D$ and m proves $\varphi(\underline{n})$,

where $\underline{\mathrm{n}}$ is the numeral for n.

When the verifying objects are partial recursive functions ϕ_e and the domain of quantification is the natural numbers $\mathbb{N}$, we obtain Kleene's notion of *recursive realizability*. For a historical account see Kleene's paper [9]. We understand by $e \Vdash \varphi$ that there is a partial recursive function with index e, which *realizes* φ. Note that $e \Vdash \neg\varphi \equiv \varphi \to \bot$ if and only if $\phi_e(n)\uparrow$ for every realiser n of φ, that is there cannot be a realiser n of φ. From this point of view, rejecting double negation elimination is a sensible choice. For suppose $e \Vdash \neg\neg\varphi$, then ϕ_e converts every proof n of $\neg\varphi$ to a proof of $\bot$, thus the domain is empty. Such a realiser n would itself convert every proof m of φ to a proof of $\bot$, again there are no such m. All we are saying is that there cannot be a construction that converts a proof of φ to a known contradiction, but this is far from an explicit construction for verifying φ itself. It simply lacks constructive content. A comprehensive list of intuitionistically valid rules can be found in [3, 5.2.1].

A topological interpretation of intuitionistic logic due to Scott [21] gives another natural reason to why double negation elimination may not necessarily hold for all objects.

Definition 2.1.16. Let Open(X) denote the lattice of open subsets of a topological space X. To each formula A assign an open set $[\![A]\!] \in \mathrm{Open}(X)$. Then define,

$$\begin{aligned}
[\![A \wedge B]\!] &= [\![A]\!] \cap [\![B]\!];\\
[\![A \vee B]\!] &= [\![A]\!] \cup [\![B]\!];\\
[\![\neg A]\!] &= \mathrm{Int}(X - [\![A]\!]);\\
[\![A \to B]\!] &= \mathrm{Int}((X - [\![A]\!]) \cup [\![B]\!]);\\
[\![\forall x.\varphi(x)]\!] &= \mathrm{Int}\left(\bigcap_{n\in D}[\![\varphi(\ulcorner n \urcorner)]\!]\right);\\
[\![\exists x.\varphi(x)]\!] &= \bigcup_{n\in D}[\![\varphi(\ulcorner n \urcorner)]\!],
\end{aligned}$$

where $\ulcorner n \urcorner$ is the name of n in the formal language and D a given domain.

It follows from the definition that $[\![\neg\neg A]\!] = \mathrm{Int}(X - [\![\neg A]\!]) = \mathrm{Int}(X - \mathrm{Int}(X - [\![A]\!])) = \mathrm{Int}(\mathrm{Cl}([\![A]\!])) \supseteq [\![A]\!]$.

2.2 Categories with an internal logic

In this section, we give a sense of how there can be an internal logic when appropriate structures are imposed on deductive systems.

Definition 2.2.1. A *category* is a deductive system satisfying

E1. $f1_A = f = 1_B f$;

E2. $h(gf) = (hg)f$,

for all $f\colon A \to B, g\colon B \to C$ and $h\colon C \to D$.

Every deductive sytem can be made into a category $\mathcal{D}/\sim$ whose objects are the formulas of $\mathcal{D}$ and whose morphisms are equivalent classes of proofs over a suitable equivalence relation, which is compatible with the axiom schema and rules of inference. For instance (ii) prevents us from differentiating between the following two proofs:

$$\dfrac{\dfrac{A \xrightarrow{f} B \qquad B \xrightarrow{g} C}{A \xrightarrow{gf} C}\,\textbf{(Cut)} \qquad C \xrightarrow{h} D}{A \xrightarrow{h(gf)} D}\,\textbf{(Cut)} \quad \sim \quad \dfrac{A \xrightarrow{f} B \qquad \dfrac{B \xrightarrow{g} C \qquad C \xrightarrow{h} D}{B \xrightarrow{hg} D}\,\textbf{(Cut)}}{A \xrightarrow{(hg)f} D}\,\textbf{(Cut)}$$

The point that proofs differ unessentially from each other is much like the fact that there may be infinitely many algorithms that implement a given function.

Definition 2.2.2. A *bicartesian closed category* is a category and an intuitionistic propositional calculus satisfying

E3. $f = \bigcirc_A$,

for all $f\colon A \to T$;

E4a. $\pi_{A,B}\langle f,\ g\rangle = f$;

E4b. $\pi'_{A,B}\langle f,\ g\rangle = g$;

E4c. $\langle \pi_{A,B}h,\ \pi'_{A,B}h\rangle = h$,

for all $f\colon C \to A, g\colon C \to B$ and $h\colon C \to A \wedge B$;

E5a. $\varepsilon_{C,B}\langle \overline{f}\pi_{A,B},\ \pi'_{A,B}\rangle = f$;

E5b. $\overline{\varepsilon_{C,B}\langle g\pi_{A,B},\ \pi'_{A,B}\rangle} = g$,

for all $f\colon A \wedge B \to C$ and $g\colon A \to (B \Rightarrow C)$;

E6. $f = \square_A$,

for all $f\colon \perp \to A$;

E7a. $[f,\ g]\kappa_{A,B} = f$;

E7b. $[f,\ g]\kappa'_{A,B} = g$;

E7c. $[h\kappa_{A,B},\ h\kappa'_{A,B}] = h$,

for all $f\colon A \to C, g\colon B \to C$ and $h\colon A \vee B \to C$.

Each calculus given by rules R3.–R7. in definition 2.1.3 give rise to a corresponding category. A conjunction calculus generates a cartesian category, equationally presented in E1.–E4. and similarly, the equations E1.–E5. present a cartesian closed category whose counterpart is a positive intuitionistic calculus. In terms of universal properties E3. for instance, expresses that $\top$ is the terminal object 1 and E5a, b. asserts the existence of exponential objects A^B such that given products and an evaluation map ε, for any $f\colon A \times B \to C$ there exists a unique transpose $\overline{f}\colon A \to C^B$ such that the following diagram commutes:

$$\begin{tikzcd}
A \arrow[d, dashed, "\overline{f}"'] & A \times B \arrow[d, dashed, "\overline{f} \times 1_B"'] \arrow[dr, "f"] & \\
C^B & C^B \times B \arrow[r, "\varepsilon_{C,B}"'] & C
\end{tikzcd} \tag{2.2.1}$$

Then E5b. ensures that the mapping $\mathcal{C}(A \times B, C) \xrightarrow{\overline{\ }} \mathcal{C}(A, C^B)$ is bijective. Here, a suitable equivalence relation would be one that for example includes the assertion $f \sim g \implies \overline{f} \sim \overline{g}$ for all proofs f and g.

It turns out that one can obtain the usual connectives, introduce quantifiers and do n-valued logic and in a bicartesian closed category. The details are given in a paper by Lambek [10]. To illustrate what constitutes an internal logic in such a category, we give an example from this paper. Define $2 = 1 + 1$ and interpret morphism $p\colon 1 \to 2$ as propositions or *truth-values*. Define specific morphism *true* $\top\colon 1 \to 2$ and *false* $\perp\colon 1 \to 2$ with $\top = \kappa_{1,1}$ and $\perp = \kappa'_{1,1}$, not to be confused with terminal and initial objects in the category. Then negation is the morphism $\neg\colon 2 \to 2$ described by $\neg = [\perp,\ \top]$ and conjunction is the morphism $\wedge\colon 2{\times}2 \to 2$ described by $\wedge = [-,\ \perp]$ so that $p \wedge q = [p,\ \perp]q$.

However, we will need more than a classical internal logic. Indeed, an in-depth explanation would require studying more broadly topos theory. For completeness we state the definition of an elementary topos according to Lawvere [12] and explore some of its elementary features below.

Definition 2.2.3. An *elementary topos* $\mathcal{E}$ is a category which

(i) is cartesian closed;

(ii) is finitely complete and cocomplete;

(iii) has a subobject classifier Ω.

Definition 2.2.4. Let C be a category and A an object of C. A *subobject* of A is an isomorphism class of monics into A. Let $Monic(A)$ be the full subcategory of C/A whose objects are monics, then $\mathrm{sub}(A)$ denotes the class of isomorphism classes of $Monic(A)$.

In **Set** subobjects of a set A are precisely its subsets. Indeed, monics $X \overset{m'}{\rightarrowtail} A$ and $Y \overset{m}{\rightarrowtail} A$ are isomorphic in $Monic(A)$ if and only if they have the same image [14, 5.1.40]. For suppose they are isomorphic, in **Set** this corresponds to having a bijection $X \xrightarrow{f} Y$ such that $mf = m'$ and $m'f^{-1} = m$ and thus $m'(X) = mf(X) = m(Y)$. For the converse define $f\colon x \mapsto m^{-1}(m(x))$ which is well-defined and bijective as m, m' have the same image and are injective. Now suppose the category C also has pullbacks, then it is easy to see that $A' \xrightarrow{f} A$ induces a morphism $\mathrm{Sub}(A) \xrightarrow{Sub(f)} \mathrm{Sub}(A); [m] \mapsto [m']$,

$$\begin{tikzcd}
S' \arrow[r, "f'"] \arrow[d, tail, "m'"'] & S \arrow[d, tail, "m"] \\
A' \arrow[r, "f"'] & A
\end{tikzcd}$$

as monics are preserved under pullback. In **Set** this just amounts to the inverse image $S' = f^{-1}(S)$. By the pasting lemma for pullbacks, we can see that $\mathrm{Sub}\colon C^{\mathrm{op}} \to \textbf{sets}$ defines a functor. If the category in addition is *well-powered*, viz. for each A we have that $\mathrm{Sub}(A)$ is small, then the category is said to have a *subobject classifier* if the functor Sub is representable [14, 6.3.26]. Namely, there is a choice of object Ω and natural isomorphism such that $\mathrm{Sub} \cong C(-, \Omega)$. Equivalently, a subobject classifier is a monic *true* $1 \overset{\top}{\rightarrowtail} \Omega$ such that for any subobject of A there exists a unique characteristic morphism $A \xrightarrow{\chi} \Omega$ such that the following following square forms a pullback [16, I.3.1].

$$\begin{tikzcd}
S \arrow[r, "\circ_S"] \arrow[d, tail, "m"'] & 1 \arrow[d, tail, "\top"] \\
A \arrow[r, dashed, "\chi"'] & \Omega
\end{tikzcd}$$

In other words, monics or rather isomorphism classes of monics into A are in one-to-one correspondence with characteristic morphisms $A \to \Omega$. The object 2 is up to isomorphism the subobject classifier in **Set** and indeed subsets S of A are in one-to-one correspondence with characteristice functions $A \xrightarrow{\chi_S} 2$.

Analogously to remark 1.1.7 we can express a predicate $R \rightarrowtail A$ via its characteristic morphism $A \xrightarrow{\chi_R} \Omega$ and take morphism $1 \overset{p}{\rightarrowtail} \Omega$ to be truth-values. We can thus derive the usual connectives, as was done above, albeit in terms of universal properties; for example

$$\begin{tikzcd}
0 \arrow[r, "\circ_0"] \arrow[d, tail, "\square_0"'] & 1 \arrow[d, tail, "\top"] & 1 \arrow[r, "\circ_1"] \arrow[d, tail, "\bot"'] & 1 \arrow[d, tail, "\top"] & \Omega \arrow[r, "\circ_\Omega"] \arrow[d, tail, "\langle\top,\top\rangle"'] & 1 \arrow[d, tail, "\top"] \\
1 \arrow[r, dashed, "\bot"'] & \Omega & \Omega \arrow[r, dashed, "\neg"'] & \Omega & \Omega\times\Omega \arrow[r, dashed, "\wedge"'] & \Omega
\end{tikzcd}$$

and forget about the archery and operate as if we were using set theory. We could therefore say that subobjects are generalisations of subsets, but the subobject classifier is a much more powerful tool than what it appears to be in **Set**.

Consider the category of graphs **Grphs**, where the objects are graphs $\mathcal{G}$ as defined in definition 2.1.1 and morphisms are functors F mapping nodes to nodes and edges to edges. Then the notion of a subobject is the familiar notion of a subgraph. Consider the red arrow in fig. 2.1 below. It is obviously not included in the $\mathcal{G}'$, yet its source and target are. In the right setting we can fine-tune our answer to account for such

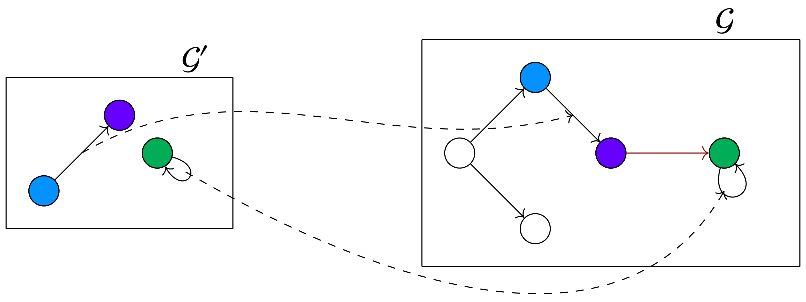

Figure 2.1: Graph $\mathcal{G}'$ is a subobject of $\mathcal{G}$ in **Grphs**.

subtleties. In **Grphs** the subobject classifier is up to isomorhpism a graph that takes into account the five cases for source, edge, target and the two cases for nodes [13], see fig. 2.2. Now, we can instead of simply asking if something *is* or *is not* true, ask more

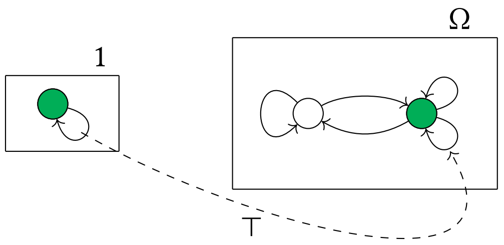

Figure 2.2: Subobject classifier in **Grphs**.

complicated and interesting questions like *where* is it true or *how true* is it. In $\mathcal{E}ff$, we even require that there be proof of a simple membership problem.

Similar to how the notion of subobject generalises subsets, the *power object* $\mathcal{P}$ generalises the notion of powerset. This is more complicated and we will not describe it here, but in the same way in which $\mathcal{P}(X) \cong 2^X$ in **Set**, internally Ω^X functions as the powerset. In $\mathcal{E}ff$, the subobject classifier is $\Omega = \mathcal{P}1$. True 1 and false 0 are its elements, but unlike the classical world these are not its only elements. Nor is it true that there are elements that differ from both 1 and 0, much like Ω in **Grphs**, except not as dicrete. Another way to describe it is $\Omega = (\mathcal{P}(\mathbb{N}), [\![p \longleftrightarrow q]\!])$ with non-standard equality $\mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N}); (p, q) \mapsto [\![p \longleftrightarrow q]\!]$ as sets, where $\longleftrightarrow$ is in the sense of Kleene in definition 2.1.15. The resulting internal logic of the effective topos is higher-order intuitionistic logic with a built-in notion of uniformly effectively in the sense of recursive realizability. This of course is a massive simplification, but good enough for the purposes of this thesis.

2.3 Partial combinatory algebras

In this section, we take a more abstract view of the models of computation and study the so called *Schönfinkel* algebras $(A, ., i, k, s)$ with a closure property crucial to logic. We establish this combinatory completeness and introduce our model of computation following van Oosten [18, Chapter 1] and Longely [15, Chapter 1].

Definition 2.3.1. A *partial applicative system* (pas) is a nonempty set A equipped with a partial binary operation $. : A \times A \rightharpoonup A$.

We call the map $(a, b) \mapsto a.b$ *application* and denote by the juxtaposition ab the result of applying a to b. As we will see, the application map is not necessarily associative. We adopt left-association and write abc for $(ab)c$ whenever unambiguous.

For elements $a, b \in A$, the term ab may not denote an element of A. The following definition provides a formal distinction between elements of A and terms over A.

Definition 2.3.2. Let A be a pas and let V be an infinite set of variables. Define the set $T(A)$ of terms over A to be the least set such that

(i) $A \subseteq T(A)$;

(ii) $V \subseteq T(A)$;

(iii) if $t \in T(A)$ and $t' \in T(A)$ then $(tt') \in T(A)$.

We understand by $t(x_1, \dots, x_n)$, the term t whose variables are among $x_1, \dots, x_n \in V$. Let t, t' be terms and x a variable. We write $t[t'/x]$ to mean the term obtained by substituting t' for x in t. We write $t[\overline{a}/\overline{x}]$ instead of $t[a_1/x, \dots, a_n/x_n]$. A term is called *closed* if no variables occur in it. We establish a relation between closed terms and elements of A.

Definition 2.3.3. For a closed term $t{\downarrow}a$, read $t \in T(A)$ is defined and denotes an element $a \in A$, is the least relation such that

(i) $a{\downarrow}a$ for all $a \in A$;

(ii) $(tt'){\downarrow}a$ if and only if there exists $b, c \in A$ such that $t{\downarrow}b$, $t'{\downarrow}c$ and $bc = a$.

If there exists an element $a \in A$ such that $t{\downarrow}a$, we simply say that t denotes and write $t{\downarrow}$. We write $t{\uparrow}$ if t is undefined. Strict equality on closed terms is such that $t = t'$ if and only if t, t' are both defined and denote the same element. Clearly, if $t{\downarrow}a$ and $t{\downarrow}b$ then $a = b$. The Kleene equality $t \simeq t'$ says that if either t or t' denote then $t = t'$. Let t, t' be terms with variables in $x_1, \dots, x_n$. Then in general, we write

$$\begin{array}{lll} t{\downarrow} & \text{if} & t[\overline{a}/\overline{x}]{\downarrow}; \\ t \simeq t' & \text{if} & t[\overline{a}/\overline{x}] \simeq t'[\overline{a}/\overline{x}] \end{array} \tag{2.3.1}$$

for all substitution instances $a_1, \dots, a_n \in A$.

Definition 2.3.4. A pas A is *combinatory complete* if for any $n \in \mathbb{N}$ and any term $t(x_1, \dots, x_{n+1})$ there exist an $a \in A$ such that for all $a_1, \dots, a_{n+1} \in A$

(i) $aa_1 ... a_n{\downarrow}$;

(ii) $aa_1 ... a_{n+1} \simeq t[\overline{a}/\overline{x}]$.

A pas A is called a *partial combinatory algebra* (pca) if A is combinatory complete.

Proposition 2.3.5. *If A is a pca then there exist element $s, k, i \in A$ such that for all $a, b, c \in A$*

(i) $kab = a$

(ii) $sab{\downarrow}$

(iii) $sabc \simeq ac(bc)$.

(iv) $ia = a$

Proof. Suppose A is combinatory complete. Take for k, s and i an element of A satisfying the conditions of definition 2.3.4 for the terms $t(x_1, x_2) \equiv x_1$, $t(x_1, x_2, x_3) \equiv x_1x_3(x_2x_3)$ and $t(x_1) \equiv x_1$ respectively. Then we have that for all $a, b \in A$ $kab \simeq a$, but a always denotes. Conditions (ii), (iii) and (iv) follow in much the same way. ■

It is entirely natural to require an identity element, albeit (iv) in proposition 2.3.5 is superfluous. Indeed i and skk are extensionally equal as for all $a \in A$, $skka \simeq ka(ka) = a$ and $a{\downarrow}$ always. In fact, one can take skX for an arbitrary $X \in A$ as long as $Xa{\downarrow}$ [20].

Typed combinatory logic corresponds to Hilbert-style axiomatic system. Curiously, this gives another natural reason to choose k. For regard

$$\begin{aligned} Ak &: (\varphi \implies (\psi \implies \varphi)); \\ As &: (\varphi \implies (\psi \implies \chi)) \implies ((\varphi \implies \psi) \implies (\varphi \implies \chi)); \end{aligned}$$

as axiom schemes and function application as modus ponens, then skk corresponds exactly to a proof of an instance of the identity with

$$\begin{aligned} s &: (A \implies ((A \implies B) \implies A)) \implies ((A \implies (A \implies B)) \implies (A \implies A)); \\ k &: (A \implies ((A \implies B) \implies A)); \\ k &: ((A \implies (A \implies B)); \\ skk &: (A \implies A). \end{aligned}$$

From now on we will use i and skk interchangeably.

Lemma 2.3.6. *Properties of s and k extend to corresponding facts about terms just when terms denote.*

Proof. Follows from definition of s, k and (2.3.1). ■

Lemma 2.3.7. *Suppose A satisfies the conditions of the proposition 2.3.5. For any term $t \in T(A)$ there exist a term $\Lambda x.t \in T(A)$ whose variables are those of t excluding x such that $(\Lambda x.t)\downarrow$ and $(\Lambda x.t)a \simeq t[a/x]$ for all $a \in A$.*

Proof. Define for every $x \in V$ and every $t \in T(A)$ a term $\Lambda x.t$ inductively on the structure of t as follows:

(i) $\Lambda x.t \equiv kt$ if t is a constant, $a \in A$ or a variable different from x;

(ii) $\Lambda x.x \equiv i$;

(iii) $\Lambda x.tt' \equiv s(\Lambda x.t)(\Lambda x.t')$.

The base case follows immediately by definition and the conditions of the proposition. For the inductive step, suppose $(\Lambda x.t)\downarrow$, $(\Lambda x.t')\downarrow$ and $(\Lambda x.t)a \simeq t[a/x]$, $(\Lambda x.t')a \simeq t'[a/x]$. Then it follows by lemma 2.3.6 that

$$\Lambda x.tt' \equiv s(\Lambda x.t)(\Lambda x.t')\downarrow;$$

$$(\Lambda x.tt')a \equiv s(\Lambda x.t)(\Lambda x.t')a \simeq (\Lambda x.t)a(\Lambda x.t')a \simeq t[a/x]t'[a/x] \equiv tt'[a/x].$$ ■

Lemma 2.3.8. *If $t'\downarrow$ and x is not among the variables of t', then $(\Lambda x.t)t' \simeq t[t'/x]$.*

Proof. Follows by structural induction on t and lemma 2.3.6. ■

Remark 2.3.9. Notice that Λ is merely a meta-syntactic sugar. It is itself not part of the formal expressions, $T(A)$. Λ-abstraction is used as a way to avoid writing long terms involving k and s. We write $\Lambda xy.t$ to abbreviate $\Lambda x.(\Lambda y.t)$. This can formally be done by first translating $\Lambda y.t$ to a term t', then taking $\Lambda x.t'$ to a term t''.

On the level of substitution, Λ-terms and terms may not agree, viz. that $(\Lambda y.t)[t'/x]$ and $\Lambda y.t[t'/x]$ are different, even if the pca is total. For instance, take $t \equiv x$ and $t' \equiv ss$, then while $(\Lambda y.x)[ss/x] \simeq k(ss)$, we have that $\Lambda y.x[ss/x] \simeq s(ks)(ks)$. The point is that Λ distinguishes between the term and the constant the term denotes. It is the case, however, that Λ is functorial with respect to substitution for constants and unbounded variables, as long as we stay in the realm of Λ-terms. That is, given a term with variables in x and y, by construction we get that

$$\Lambda y.t[a/x] = (\Lambda y.t)[a/x]. \tag{2.3.2}$$

just when $\Lambda y.t[a/x]$ denotes, but by lemma 2.3.7 it always does.

We will use lambda abstraction readily as it allows us to go to the level of terms which always denote regardless of the terms denoting.

Lemma 2.3.10. *Given a term $t(x_1, \dots, x_r, x_{r+1}, \dots, x_{n+1})$, there exists a denoting term $\hat{t} \equiv \Lambda x_{r+1}...x_{n+1}.t$ with variables in $x_1, \dots, x_r$ such that $(\Lambda x_1...x_r.\hat{t})a_1...a_r$ denotes and*

$$((\Lambda x_1...x_r.\hat{t})a_1...a_r)a_{r+1} \dots a_{n+1} \simeq t[\overline{a}/\overline{x}].$$

Proof. By our construction in remark 2.3.9 and lemma 2.3.7 it follows that the term $\hat{t}$ with variables in $x_1, \dots, x_r$ denotes. Again, by repeated application of lemma 2.3.7 and (2.3.2) we get that

$$\begin{aligned}
(\Lambda x_1...x_r.\hat{t})a_1...a_r &\simeq ((\Lambda x_2...x_r.\hat{t})[a_1/x_1])a_2...a_r \\
&= (\Lambda x_2...x_r.\hat{t}[a_1/x_1])a_2...a_r \\
&\vdots \\
&\simeq \hat{t}[a_1/x_1, \dots, a_r/x_r] \\
&\equiv (\Lambda x_{r+1}...x_{n+1}.t)[a_1/x_1, \dots, a_r/x_r] \\
&= \Lambda x_{r+1}...x_{n+1}.t[a_1/x_1, \dots, a_r/x_r]
\end{aligned}$$

denotes and thus

$$\begin{aligned}
((\Lambda x_1...x_r.\hat{t})a_1...a_r)a_{r+1} \dots a_{n+1} &= (\Lambda x_{r+1}...x_{n+1}.t[a_1/x_1, \dots, a_r/x_r])a_{r+1} \dots a_{n+1} \\
&\simeq t[\overline{a}/\overline{x}].
\end{aligned}$$

■

Proposition 2.3.11 (Feferman). *Let A be a pas. Then A is a pca if there exist elements $s, k \in A$ satisfying the properties (i)–(iii) in 2.3.5.*

Proof. Take $\hat{t} \equiv \Lambda x_1 ... x_{n+1}.t$, say with no free variable. Hence all these Λ-abstractions must denote an element, now that all the free variables are exhausted. It follows by lemma 2.3.10 that it is precisely this element $\hat{t}$ we want for the converse of proposition 2.3.5. ■

Proposition 2.3.12. *There exist constants $p, p_0, p_1 \in A$ such that for all $a, b \in A$,*

$$pab{\downarrow}, \qquad p_0(pab) = a, \qquad p_1(pab) = b.$$

Proof. Let $p \equiv \Lambda xyz.zxy$. It is clear by lemma 2.3.7 and (2.3.2) that $pab \simeq \Lambda z.zab$ denotes. Have $\bar{k}$ denote ki so that $\bar{k}ab = kiab = ib = b$. Then define $p_0 \equiv \Lambda v.vk$ and $p_1 \equiv \Lambda v.v\bar{k}$. It follows by lemma 2.3.8 that

$$\begin{aligned} p_0(pab) &\simeq pabk \simeq (\Lambda z.zab)k \simeq kab = a; \\ p_1(pab) &\simeq pab\bar{k} \simeq (\Lambda z.zab)\bar{k} \simeq \bar{k}ab = b \end{aligned}$$

for all $a, b \in A$. ■

We can therefore take p, p_0 and p_1 as a code for our *pairing* and *projection* operators. We can further take k or likewise $\Lambda yz.y$ and $\bar{k}$ or likewise $\Lambda yz.z$ to act as our Booleans, *true* and *false*. From here we can create 'if-else' statements by identifying an element *if* by $\Lambda xyz.xyz$ with the property that for all $a, b \in A$ $if ab{\downarrow}$, $if true ab = a$ and $if false ab = b$. We can also simulate the natural numbers in a pca.

Definition 2.3.13. The *Curry numerals* in a pca A are defined as follows:

$$\begin{aligned} \bar{0} &\equiv pki; \\ \overline{n+1} &\equiv p\bar{k}\bar{n}, \end{aligned}$$

where p is the pairing operator.

The *succ* operator is thus described by $\Lambda x.p\bar{k}x$ and the first projection, p_0 simply gives a code for an *iszero* test. From which, we obtain a code $\Lambda x.if(iszero\, x)\bar{0}(p_1 x)$ for the *pred* operator.

Proposition 2.3.14 (Fixed point operators). *There exist elements $y, z \in A$ such that for all $a, f \in A$,*

$$yf \simeq f(yf), \qquad zf{\downarrow}, \qquad (zf)a \simeq f(zf)a.$$

Proof. Let $w \equiv \Lambda xu.u(xxu)$ and define $y \equiv ww$, then we get the desired result:

$$yf \equiv wwf \equiv (\Lambda xu.u(xxu))wf \simeq (\Lambda u.u(wwu))f \simeq f(wf).$$

Similarly, let $v \equiv \Lambda xyu.y(xxy)u$ and $z \equiv vv$ so that

$$zf \equiv (\Lambda xyu.y(xxy)u)vf \simeq (\Lambda yu.y(vvy)u)f \simeq \Lambda u.f(zf)u,$$

which always denotes. By another application of lemma 2.3.7 we have

$$(zf)a \simeq (\Lambda u.f(zf)u)a \simeq f(zf) \blacksquare$$

Proposition 2.3.15. *The primitive recursive operator rec is in A with the property that for all $a, f \in A$*

$$rec\ a\ f\ \overline{0} = a, \qquad rec\ a\ f\ \overline{n+1} \simeq f\ \overline{n}\ (rec\ af\ \ \overline{n}).$$

Proof. Let $R \equiv \Lambda rxfm.if(iszero\ m)(kx)(\Lambda y.f(pred\ m)(rxf(pred\ m)i))$ and use the fixed point operator z to define $rec \equiv \Lambda xfm.(zR)xfmi$. We get,

$$\begin{aligned}
rec\ af\ \overline{0} &\equiv (\Lambda xfm.(zR)xfmi)af\ \overline{0} \\
&\simeq (zR)af\ \overline{0}\ i \\
&\simeq R(zR)af\ \overline{0}\ i \\
&\simeq if(iszero\ \overline{0})(ka)(\Lambda y.f(pred\ \overline{0})((zR)af(pred\ \overline{0})i))i \\
&\simeq kai = a;
\end{aligned}$$

and

$$\begin{aligned}
rec\ af\ \overline{n+1} &\equiv (\Lambda xfm.(zR)xfmi)af\ \overline{n+1} \\
&\simeq (zR)af\ \overline{n+1}\ i \\
&\simeq R(zR)af\ \overline{n+1}\ i \\
&\simeq if(iszero\ \overline{n+1})(ka)(\Lambda y.f(pred\ \overline{n+1})((zR)af(pred\ \overline{n+1})i))i \\
&\simeq (\Lambda y.f(pred\ \overline{n+1})((zR)af(pred\ \overline{n+1})i))i \\
&\simeq f\overline{n}((zR)af\overline{n}i) \\
&\simeq f\overline{n}(rec\ af\ \overline{n})
\end{aligned}$$

as desired. ■

Addition can now be easily expressed as $add \equiv \Lambda xy.rec\ x\ (k\ succ)\ y$. There is nothing inherently unique about the representation of the operators we have identified so far. Rather, objects are determined by their relative behaviour.

Proposition 2.3.16. *Suppose there are elements $\overline{0}', \overline{1}', \ldots \in A$ and $succ', rec' \in A$ with the properties that for all $a, f \in A$*

$$succ'\ \overline{n}' = \overline{n+1}', \qquad rec'\ af\ \overline{0}' = a, \qquad rec'\ af\ \overline{n+1}' \simeq f\ \overline{n}'(rec'\ af\ \overline{n}').$$

Then there exists elements $c, d \in A$ such that for all n $c\overline{n} = \overline{n}'$ and $d\overline{n}' = \overline{n}$.

Proof. The elements $c \equiv rec\ \overline{0}'(k\ succ)$ and $d \equiv rec'\ \overline{0}(k\ succ')$ do the job. ∎

Proposition 2.3.17. *A pca A is trivial if and on if*

(i) *the application map is associative;*

(ii) *the application map is commutative;*

(iii) $k = s$.

Proof. For (i), let $A = \{ a \}$ be a trivial pca. Then either $aa{\downarrow}a$ or $aa{\uparrow}$. In either case, $(aa)a \simeq a(aa)$. Conversely, associativity would imply that $k = kkk = k(kk)$ so that for all $a \in A$, $a = kak = k(kk)ak = k$.

Commutativity is evident for (ii) in the case that A is trivial. It follows by commutativity that $skk = ksk = s$ so that for all $a \in A$, $a = skka = (skk)kka = kka = k$.

For part (iii) it is obvious that $k = s$ if the pca is trivial. If $k = s$, then $skk = kkk = k$ from which it follows that for all $a \in A$, $a = skka = skk(skk)a = kka = k$. ∎

Proposition 2.3.18. *Suppose A is non-trivial and that the application map is not total. Then there exist an element $e \in A$ such that for all $a \in A$, $ea{\uparrow}$.*

Proof. As the application map is not total, there exist elements $b, c \in A$ such that $bc{\uparrow}$. Then $\Lambda x.bc$ does the job: for all $a \in A$, $(\Lambda x.bc)a \equiv s(kb)(kc)a \simeq kba(kca) = bc$. ∎

Recognise that this is the everywhere divergent function. Our model of computation will be Kleene's first model $\mathcal{K}_1$ in which we take $A = \mathbb{N}$ and application $e.n \equiv \phi_e(n)$, where the ϕ_e are precisely the partial recursive functions we studied in Chapter 1.

2.4 Category of assemblies and modest sets

In computers every datatype has a binary representation, but we might as well have taken a different representation such as the natural numbers. Here, we use the Schönfinkel algebras as abstract machines on which various datatypes are implemented. We will see that these form a bicartesian category with non-standard truth-values. Our references are again from [18, 15].

Definition 2.4.1. Let A be a pca. An *A-valued assembly* X is a set $|X|$ together with a function $E\colon |X| \to \mathcal{P}^*(A)$ assigning to each $x \in X$ a nonempty subset Ex of A.

In the setting of recursive realizability, we think of Ex as the set of proofs for x.

Definition 2.4.2. Suppose $(|X|, E)$, $(|Y|, F)$ are two A-valued assemblies. A function $f\colon |X| \to |Y|$ is said to be *tracked* by an element $t \in A$ if for all $x \in X$ and for all $a \in Ex$, $ta{\downarrow}$ and $ta \in Ff(x)$.

Proposition 2.4.3. *Assemblies on A form a category $Ass(A)$.*

Proof. Let a *morphism* $f\colon X \to Y$ of assemblies be a function $f\colon |X| \to |Y|$ that is tracked by some element $t \in A$. Clearly, i tracks $1_X\colon X \to X$. Suppose s tracks $f\colon X \to Y$ and t tracks $g\colon Y \to Z$, then $\Lambda x.t(sx)$ tracks their composition. That assmeblies and their morphisms form a catgeory follows now from the underlying function satisfying the identity laws and associativity. ■

By abuse of notation we will sometimes refer to the assembly and the underlying set, and the morphism and the underlying function by the same name. Assemblies are one way of categorising pcas [18, 1.2.1]. They can intuitively be thought as data types with an underlying set of values $|X|$ whose elements are given machine-level representations, or in our setting, realisers Ex [15]. Then morphisms between assemblies are precisely the functions that can be simulated, in our case, by a partial recursive function acting on the realisers instead of elements.

Proposition 2.4.4. *Ass*(A) *is cartesian closed.*

Proof. The terminal object 1 is given by the assembly $(|\{ * \}|, T)$, where $T* \equiv A$ and every morphism into 1 is tracked by i. Products of assemblies X and Y are given by $(|X| \times |Y|, P)$, with $Px,y \equiv \{\, pab \mid a \in Ex, b \in Fy \,\}$. It is easy to see that the following diagram commutes and that the construction is unique.

$$\begin{tikzcd}
 & & (|X|, E) \\
(|Z|, G) \arrow[rru, "s \Vdash f_1"] \arrow[r, dashed, "\Lambda x.p(sx)(tx) \Vdash \langle f_1, f_2\rangle"] \arrow[rrd, "t \Vdash f_2"'] & (|X| \times |Y|, P) \arrow[ru, "p_0 \Vdash \pi"'] \arrow[rd, "p_1 \Vdash \pi'"] & \\
 & & (|Y|, F)
\end{tikzcd}$$

It remains to show that the functor $- \times Y\colon$ *Ass*$(A) \to$ *Ass*(A) has a right adjoint $(-)^Y$. Let exponential objects be given by

$$|Z^Y| \equiv \{\, f\colon |Y| \to |Z| \mid \text{f is tracked by some } t \in A \,\}$$
$$Hf \equiv \{\, t \in A \mid t \text{ tracks } f \,\}.$$

Suppose t tracks $f\colon |X| \times |Y| \to |Z|$, then $\Lambda xy.t(pxy)$ tracks the exponential transpose $\overline{f}$. Similarly, if s tracks $g : |X| \to |Z^Y|$ then $\Lambda x.s(p_0x)(p_1x)$ tracks $\overline{\mathrm{g}}$. In particular, the evaluation map $\varepsilon : X^Y \times Y \to X$ is realised by $\Lambda x.(p_0x)(p_1x)$. We show that the universal property (2.2.1) is satisfied. Given $f : |X| \times |Y| \to |Z|$ define $\overline{f}$ such that $\overline{f}(x)(y) = f(x,y)$ so that $\varepsilon(\langle \overline{f}\pi, \pi'\rangle)(x,y) = \varepsilon(\langle \overline{f}(x), y\rangle) = \overline{f}(x)(y) = f(x,y)$ as desired. Also, $\overline{\varepsilon(\langle \overline{\mathrm{g}}\pi, \pi'\rangle)}(x)(y) = \varepsilon(\langle \overline{\mathrm{g}}\pi, \pi'\rangle)(x,y) = g(x)(y)$. ■

Remark 2.4.5. Indeed, terminal objects are unique up to a unique isomorphism, but what is interesting is the choice of $T*$. It can be any nonempty subset of A with the unique map tracked by a constant operator $\Lambda x.a$ for some $a \in T*$.

The representations of the elements in the underlying set is not unique as the set of realisers are not necessarily disjoint. We therefore ask for those datatypes for which the codes uniquely determine each value. The following definition captures this idea.

Definition 2.4.6. An assembly is said to be a *modest set* if for all $x, x' \in |X|$,

$$x \neq x' \implies Ex \cap Ex' = \emptyset.$$

Let $Mod(A)$ denote the category of modest sets. It is easy to see that it is a full subcategory of $Ass(A)$, for the morphisms between two modest sets are precisely those that are tracked. The following lemma states that Y^X is modest whenever Y is modest.

Lemma 2.4.7. *Suppose $t \Vdash f, f' \colon X \to Y$ and Y modest, then $f = f'$.*

Proof. We have that for all $x \in X$ and for all $a \in Ex$, $ta\downarrow$ with $ta \in Ff(x)$ and $ta \in Ff'(x)$ so that $Ff(x) \cap Ff'(x) \neq \emptyset$. It follows that $f(x) = f'(x)$ for all x as Y is modest, hence $f = f'$. ■

Proposition 2.4.8. *The category $Mod(A)$ is cartesian closed and the obvious inclusion $J \colon Mod(A) \hookrightarrow Ass(A)$ preserves the structure.*

Proof. The terminal object 1 is clearly modest. Let X, Y be modest sets, then $X \times Y$ is modest. For suppose $r \in Px, y \cap Px', y'$ then $p_0 r \in Ex \cap Ex'$ and $p_1 r \in Ey \cap Ey'$ so that $x = x'$ and $y = y'$. It follows from lemma 2.4.7 that $Mod(A)$ is cartesian closed. ■

Proposition 2.4.9. *Both $Ass(A)$ and $Mod(A)$ are finitely complete (and finite limits are preseved under J).*

Proof. It suffices to show that $Ass(A)$ has equalizers [14, 5.1.38]. Given morphisms $f, g \colon X \to Y$, take $X' = \{ x \in X \mid f(x) = g(x) \}$ and let E' be the restriction of X to X' so that $E'x = Ex$ with $i \Vdash e \colon X' \to X$. It is clear from the definition that the equaliser X' is modest whenever X is. ■

Proposition 2.4.10. *Both $Ass(A)$ and $Mod(A)$ are bicartesian closed and finitely cocomplete (and J preserves the structure).*

Proof. The initial object 0 is given by $(\emptyset, \emptyset)$ which is trivially modest. The coproduct of two assemblies is the object with $\{ 0 \} \times |X| \cup \{ 1 \} \times |Y|$ as the underlying set and existence $Gz \equiv \{ p\bar{0}a \mid a \in E\pi'(z) \} \cup \{ p\bar{1}b \mid b \in F\pi'(z) \}$. The coproduct is modest whenever both are by definition. The universal property is easily checked from the following diagram.

$$\begin{tikzcd}
(|X|, E) \arrow[dr, "\Lambda x.p\overline{0}x \Vdash \kappa"'] \arrow[drr, "t \Vdash g_1"] & & \\
& (|X| + |Y|, G) \arrow[r, dashed, "\Lambda u.if\ (iszero\, u)(t(p_1 u))(s(p_1 u)) \Vdash [g_1, g_2]"] & (|Z|, Q) \\
(|Y|, F) \arrow[ur, "\Lambda y.p\overline{1}y \Vdash \kappa'"] \arrow[urr, "s \Vdash g_2"'] & &
\end{tikzcd}$$

For finite colimits, it dually suffices to show that we have coequalizers. Given the maps $f, g\colon (|X|, E) \to (|Y|, F)$ the coequalizers is given by $(|Y|/{\sim}, F')$, where $\sim$ is the equivalence relation $\{\, (f(x), g(x)) \mid x \in |X| \,\} = \{\, (y, y') \mid \exists x \in X(f(x) = y \wedge g(x) = y') \,\}$ and $F'[y] \equiv \bigcup\limits_{y \sim y'} Fy$. Then $i \Vdash p\colon (|Y|, F) \to (|Y|/{\sim}, F')$. Note that the quotient is modest whenever Y is, for $[y] \neq [y']$ then $y \nsim y'$ and so $F'[y] \cap F'[y'] = \emptyset$. ∎

Proposition 2.4.11. *Mod*(A) *and Ass*(A) *have a natural number object (and it is preserved under J).*

Proof. Given the modest set $N \equiv (\mathbb{N}, E)$ with $En \equiv \{\, \overline{n} \,\}$ and morphisms $1 \xrightarrow{0} N, N \xrightarrow{s} N$ tracked by $\Lambda x.\overline{0}$ and $succ$ respectively, if Y is an assembly with morphisms $1 \xrightarrow{y} Y, Y \xrightarrow{f} Y$ then there is a unique morphism $N \xrightarrow{x} Y$ such that

$$\begin{tikzcd}
& N \arrow[r, "succ \Vdash s"] \arrow[d, dashed, "rec(ri)(kt) \Vdash x"] & N \arrow[d, dashed, "rec(ri)(kt) \Vdash x"] \\
1 \arrow[ur, "\Lambda x.\overline{0} \Vdash 0"] \arrow[r, "r \Vdash y"'] & Y \arrow[r, "t \Vdash f"'] & Y
\end{tikzcd}$$

commutes. For define x recursively by $x(0) = y, x(s(n)) = f(x(n))$. We then have that $rec(ri)(kt)\overline{0} \simeq ri$, with $i \in A$ so that $ri\downarrow$ and $ri \in Fy$. Now suppose $rec(ri)(ki)\overline{n} \in Fy'$ with $f(x(n)) = y'$, then $rec(ri)(ki)\overline{n+1} \simeq t(rec(ri)(ki)\overline{n})$ denotes and is in $Ff(y')$. ∎

In $Ass(\mathcal{K}_1)$ and $Mod(\mathcal{K}_1)$ we can take the natural numbers to represent themselves. The internal language of a cartesian closed category is simply typed λ-calclulus, where the objects of the category A serve as basic *types* and morphisms as basic *terms* [11]. We also have product types $A \times B$ and the internal homs B^A serve as function types. For $1 \xrightarrow{x} A$ we write $x : A$ to mean x has type A. Then if $f : B^A$ and $a : A$, in terms of (2.1.5) $f`a : B$ is just internal function application. We will use a suitable internal language without much reference hereafter.

Chapter 3

Synthetic Computability Theory

The first steps in synthetic computability theory in the effective topos have been taken by Bauer [1]. In this chapter, we take a few extra steps in this direction. Now the following is a nice fact: modest sets $\mathit{Mod}(\mathcal{K}_1)$ can be regarded as a category internal to assemblies $\mathit{Ass}(\mathcal{K}_1)$ which is internally complete [6]. For what this kind of internalization means in a more general context see [4]. We will use this fact in order to carry on our investigation in $\mathit{Mod}(\mathcal{K}_1)$ and $\mathit{Ass}(\mathcal{K}_1)$.

3.1 Preliminaries

While Ω itself is not an object of $\mathit{Ass}(\mathcal{K}_1)$, two of its subobjects of interest 2 and Σ are. We explore their basic properties in this section.

Definition 3.1.1. The set of *decidable* truth-values is described as

$$2 \equiv \{\, p \in \Omega \mid p \vee \neg p \,\}.$$

The morphism $2 \rightarrowtail \Omega$ is a subobject of Ω and $p \in \Omega$ refers to the global elements $1 \overset{p}{\rightarrowtail} \Omega$ regarded as truth-values. Here, these are precisely the truth-values that satisfy the law of excluded middle. Up to isomorphism it is the set $(\{1,0\}, E)$, where $E1 = \{1\}$ and $E0 = \{0\}$, which is clearly modest [18, 3.2.7]. The object 2 is called the *decidable subobject classifier* and indeed there is a one-to-one correspondence between the decidable subobjects of X and morphisms $X \to 2$. In particular, 2^N is the object of decidable subobjects of N. Recall that these are precisely the subsets of $\mathbb{N}$ that posess a recursive characteristic function.

Definition 3.1.2. The Cantor space 2^N is described by the assembly (R, E) with

$$\begin{aligned} R &\equiv \{\, f \colon \mathbb{N} \to 2 \mid f \text{ is recurisve} \,\} \\ Ef &\equiv \{\, e \mid e \text{ is Gödel number for } f \,\}. \end{aligned}$$

Definition 3.1.3. The space of functions N^N is described by the assembly (Rec, E) with

$$\begin{aligned} Rec &\equiv \{\, f\colon \mathbb{N} \to \mathbb{N} \mid f \text{ is recursive} \,\} \\ Ef &\equiv \{\, e \mid e \text{ is Gödel number for } f \,\}. \end{aligned}$$

Fact 3.1.4. *[18, 3.2.26] In $\mathcal{E}ff$, the objects 2^N and N^N are isomorphic.*

Definition 3.1.5. The set of *semidecidable* truth-values is described as

$$\Sigma \equiv \{\, p \in \Omega \mid \exists f\!:\!N^N(p \leftrightarrow (\exists n(f(n) = 0))) \,\}.$$

The category of partial functions over sets **Ptl(Set)** is quivalent to the category of pointed sets $\mathbf{Set}_\perp$ with a distinguished element $\perp$, under the canonical mapping

$$A \mapsto A_\perp = A \sqcup \{\, \perp \,\}; \quad \text{(disjoint union)} \tag{3.1.1}$$

$$S \xrightarrow{(S,f)} B \mapsto A_\perp \xrightarrow{f_\perp} B_\perp; \quad a \mapsto \begin{cases} f(a) \text{ if } a \in S; \\ \perp \text{ otherwise,} \end{cases} \tag{3.1.2}$$

where $S \subseteq A$ [14, 2.3.12]. This is part of a more general construction called lifting monads [2]. In the effective topos $\Sigma = 1_\perp$, here however, $\perp$ gives an undefined element that is not-so-distinguished from the others. We can understand this peculiarity based on our intuition from the classical world. Recall that K denotes the diagonal halting set, which possesses a partial recursice characteristic function. The following fact shows that truth and falsehood are not quite seperated.

Fact 3.1.6. *[18, 3.2.27] In $\mathcal{E}ff$, the object Σ is up to isomorphism the assembly $(\{\, 1, 0 \,\}, E)$ with $E1 = K$ and $E0 = \overline{K}$.*

In technical terms, truth and falsehood are recursively inseperable, that is there is no recursive set $C \subseteq \mathbb{N}$ such that $K \subseteq C$ and $\overline{K} \subseteq \overline{C}$. A good intuition for what goes on in the $\mathcal{E}ff$ is to revisit the topological view and consider the analogous object to Σ in the category of topological spaces **Top**. There the Sierpinski space S, the space with two points and three open sets, plays the role of the object Σ. For in the same way that there is a one-to-one correspondence between continous maps from any topological space $X \to S$ and open subsets of X in **Top** [14, 4.1.30], there is a one-to-one correspondence between semidecidable subobjects of N and tracked maps $N \to \Sigma$. The subobject $\Sigma \rightarrowtail \Omega$ is called the *semidecidable* or *r.e subobject classifier* because of the following reason.

Fact 3.1.7. *[18, 3.2.28] The object Σ^N is isomorphic to the assembly (RE, W) with*

$$\begin{aligned} RE &\equiv \{\, R \subseteq \mathbb{N} \mid R \textit{ is recursively enumerable} \,\} \\ WR &\equiv \{\, e \mid R = W_e \,\}. \end{aligned}$$

The analogy is pretty good and indeed we could consider Σ^N as the lattice of 'open' sets of N, albeit r.e subsets are only closed under countable join, see proposition 1.3.18. Many of the p.r functions informally defined in Chapter 1 had a similar form to the mapping in eq. (3.1.2), but unlike there, the membership problem was semidecidable. The Σ-partial functions $N \to N_\perp$ are the synthetic analogue of partial recursive functions in the effective topos whose domains are precisly the semidecidable subobjects of N, for details see [1, 4]. It is now clear that $2 \subseteq \Sigma \subseteq \Omega$ with equality in **Set**, and thus $2^N \subseteq \Sigma^N$.

Definition 3.1.8. An object P in a topos $\mathcal{E}$ is called *internally projective* if the exponential object functor $(-)^P \colon \mathcal{E} \to \mathcal{E}$ preserves epis. That is, given an epimorphism $X \overset{f}{\twoheadrightarrow} Y$ then $X^P \overset{f^P}{\to} Y^P$ is also epic.

Definition 3.1.9. Let $\mathcal{C}$ be a locally small category. An object P is said to be *projective* if the hom-functor $\mathcal{C}(P, -) \colon \mathcal{C} \to$ **Set** preserves epis.

Fact 3.1.10. *[18, 3.2.3] The terminal object* 1 *in $\mathcal{E}ff$ is projective and the natural number object N in $\mathcal{E}ff$ is internally projective.*

Putting the above two facts together we can deduce the following neat result.

Corollary 3.1.11. *The axiom of countable choice (ACC),*

$$\forall n{:}N \exists x{:}X R(n, x) \to \exists \alpha{:}(X^N) \forall n{:}N R(n, \alpha(n))$$

holds for any object X in $\mathcal{E}ff$.

Proof. Given an epimorphism $X \overset{f}{\twoheadrightarrow} N$ then we have that $X^N \overset{f}{\twoheadrightarrow} N^N$ is also epic. Now as $\mathcal{E}ff(1, -)$ preserves epis we have that $\mathcal{E}ff(1, X^N) \to \mathcal{E}ff(1, N^N)$ is surjective and thus there exists a morphism $N \overset{g}{\to} X$ such that $(N \overset{g}{\to} X \overset{f}{\to} N) = (N \overset{1_N}{\to} N)$. That is, every epimorphism to N splits. Given a relation R, the above internally reads as the desired statement. ■

Fact 3.1.12. *We take for granted a pairing and an unpairing*

$$N \times N \to N; (a, b) \mapsto \langle a,\ b\rangle,$$
$$N \to N \times N; n \mapsto (n_0, n_1),$$

that are an isomorphism.

Fact 3.1.13. *There exists an enumeration $\phi \colon N \twoheadrightarrow N_\perp^N$ such that $\forall \psi{:}N_\perp^N \exists e{:}N \phi(e) = \psi$.*

Using pairing we get an enumeration $\phi_2 \colon N \twoheadrightarrow N_\perp^{(N^2)}$ such that $\phi_2(e)(a, b) = \phi(e)(\langle a, b\rangle)$. We can continue the pattern to get a epimorphism ϕ_k for any natural number k.

Fact 3.1.14. *There exists an enumeration $W \colon N \twoheadrightarrow \Sigma^N$ such that $\forall A{:}\Sigma^N \exists e{:}N W_e = A$.*

3.2 Basic synthetic results

The various results in the two coming sections emerged in discussion with Professor Martin Hyland. Unless otherwise stated, to the best of our knowledge these results have not appeared in the literature.

Theorem 3.2.1. *In $\mathcal{E}ff$, the s-m-n theorem holds:*

$$\exists s_n^m : N^{(N^{m+1})} \forall e, y_1, \dots, y_m : N \lambda \overline{x}.\phi_{m+n}(e)(\overline{y}, \overline{x}) = \phi_n(s(e, \overline{y}))$$

Proof. We give a proof for the case s_1^1, the general case follows by pairing. Take $e, y : N$ so that $\lambda x.\phi_2(e)(y, x) : N_\perp^N$. By fact 3.1.12 and 3.1.13,

$$\forall (e, y) : N^2 \exists d : N \lambda x.\phi_2(e)(y, x) = \phi(d).$$

Then the desired result follows by ACC ,

$$\exists s : N^{(N^2)} \forall (e, y) : N^2 \lambda x.\phi_2(e)(y, x) = \phi(s(e, y)). \qquad \blacksquare$$

Theorem 3.2.2. *In $\mathcal{E}ff$, the Fixed point theorem holds cf. [1, 4.23]:*

$$\forall f : N^N \exists n : N \phi(f(n)) = \phi(n)$$

Proof. Take $f : N^N$, which gives a map $\lambda u.f(s(u, u)) \colon N \to N$. Composing with ϕ we get $\lambda u.\phi(f(s(u, u))) \colon N \to N_\perp^N$ or equivalently a map $\lambda ux.\phi(f(s(u, u)))(x) \colon N^2 \to N_\perp$. By surjection of ϕ_2, there exists a $v : N$ such that for all $u, x : N$,

$$\phi_2(v)(u, x) = \phi(f(s(u, u)))(x). \tag{3.2.1}$$

Now let $n = s(v, v)$, then for all $x : N$

$$\begin{aligned} \phi(f(n))(x) &= \phi(f(s(v, v)))(x) \\ &= \phi_2(v)(v, x) && \text{by eq. (3.2.1)} \\ &= \phi(s(v, v))(x) && \text{by theorem 3.2.1} \\ &= \phi(n)(x). \end{aligned}$$

$\blacksquare$

Theorem 3.2.3. *In $\mathcal{E}ff$, the Second recursion theorem holds:*

$$\forall f : N^{(N^2)} \exists n : N^N \forall x : N \phi(f(n(x), x)) = \phi(n(x))$$

Proof. Take $f : N^{(N^2)}$, which gives a map $\lambda ux.f(s_1^2(u, u, x), x) : N^2 \to N$. Then we compose with ϕ to get a map $\lambda ux.\phi(f(s_1^2(u, u, x), x)) \colon N^2 \to N_\perp^N$ or equivalently a map

$\lambda uxy.\phi(f(s^2_1(u,u,x),x))(y)\colon N^3 \to N_\perp$. By surjection of ϕ_3, there exists a $v{:}N$ such that for all $u,x,y{:}N$,

$$\phi_2(v)(u,x,y) = \phi(f(s^2_1(u,u,x),x))(y). \tag{3.2.2}$$

Now let $n = \lambda x.s^2_1(v,v,x)$, then for all $y{:}N$

$$\begin{aligned} \phi(f(n(x),x))(y) &= \phi(f(s^2_1(v,v,x),x))(y) \\ &= \phi_2(v)(v,x,y) && \text{by eq. (3.2.2)} \\ &= \phi(s^2_1(v,v,x))(y) && \text{by theorem 3.2.1} \\ &= \phi(n(x))(y). \end{aligned}$$

■

3.3 Synthetic Myhill's theorem

In this section we establish our main theorem, namely that creativeness and completeness conincide in the effective topos. We will also show that $K : \Sigma^N$ is undecidable in the strong sense that it is creative. This version is even stronger than its classical counterpart as we shall see. We begin by showing the weak version, which is well known.

Proposition 3.3.1. *In $\mathcal{E}ff$, the set K is undecidable, that is*

$$\forall R{:}2^N R \neq K.$$

Proof. The argument mimics the classical one. Suppose towards a contradiction that $K = R{:}2^N$, consider $\overline{R}{:}2^N$. By fact 3.1.14 there exists an e such that $\overline{R} = W_e$, but then

$$e{:}\overline{R} \longleftrightarrow e{:}W_e \longleftrightarrow e{:}K \longleftrightarrow e{:}R,$$

which is impossible as 2^N is the object of decidable subobjects. ■

Let us first consider Myhill's characterisation, see proposition 1.3.27 in our setting. The statement that K is creative would read as follows,

$$\forall A{:}\Sigma^N[\exists n{:}A \cap K \vee \exists n(n{:}A \cup K \to \perp)]. \tag{3.3.1}$$

Now consider $A = \varnothing$, then first assertion of the disjunction is false and the second is true, while if $A = N$ then the situation is reversed. Recall, however, that Σ truth-values are recursively inseperable. Thus the above statement is asking us to do too much 'work', and in fact the standard characterisation is not valid in $\mathcal{E}ff$.

Definition 3.3.2 (Hyland). In $\mathcal{E}ff$, a set $A{:}\Sigma^N$ is creative if $\exists u{:}N_\perp^N \forall e{:}N$

(i) $\exists n{:}W_e \cap A \quad \vee \quad u(e){:}N$;

(ii) $u(e){:}W_e \cup A \to \exists n{:}W_e \cap A$.

Remark 3.3.3. We have that N is regarded as a Σ-subobject of $N_\perp$ via the pullback

$$\begin{tikzcd} N \arrow[r] \arrow[d, tail] & 1 \arrow[d, tail] \\ N_\perp \arrow[r] & \Sigma \end{tikzcd}$$

so that $u(x){:}N$ means $u(x)\downarrow$.

Note that our definition implies the standard one and is classically equivalent to it. Constructively it is a version, which has as much constructive information as possible.

Proposition 3.3.4. *A set C is creative if and only if there exists a unary partial function u such that for all x,*

(i) $\exists n \in W_x \cap A \quad \vee \quad u(x)\downarrow$*;*

(ii) $u(x) \in W_x \cup A \implies \exists n \in W_x \cap A$*.*

Proof. It is a matter of fiddling around with the logic. Suppose C is creative, that is

$$(\forall x)[W_x \cap C = \emptyset \implies [u(x)\downarrow \quad \wedge \quad u(x) \notin W_x \cup C]];$$
$$(\forall x)[\neg[u(x)\downarrow \quad \wedge \quad u(x) \notin W_x \cup C] \implies \neg[W_x \cap C = \emptyset]];$$
$$(\forall x)[[\neg u(x)\downarrow \quad \vee \quad u(x) \in W_x \cup C] \implies \neg[\forall n(n \notin W_x \cap C)]];$$
$$(\forall x)[[\neg u(x)\downarrow \implies \exists n(\neg(n \notin W_x \cap C))] \quad \wedge \quad [u(x) \in W_x \cup C \implies \exists n(\neg(n \notin W_x \cap C))]];$$
$$(\forall x)[[u(x)\downarrow \quad \vee \quad \exists n \in W_x \cap C] \quad \wedge \quad [u(x) \in W_x \cup C \implies \exists n \in W_x \cap C]];$$

as desired.

Conversely, fix an x and suppose $W_x \cap C = \emptyset$, then $u(x)\downarrow$ by (i) and $u(x) \notin W_x \cup A$ by (ii). ■

Proposition 3.3.5. *In $\mathcal{E}ff$, the set K is creative.*

Proof. Take u in definition 3.3.2 to be the identity morphism. Then (i) is trivially realised and note for (ii) that $e{:}W_e$ if and only if $e{:}K$. ■

Note that $f^{-1}(A){:}\Sigma^N$ whenever $A{:}\Sigma^N$ with characteristic morphism $A \circ f$.

Definition 3.3.6. In $\mathcal{E}ff$, the set $A{:}\Sigma^N$ is complete if and only if $\forall B{:}\Sigma^N \exists f{:}N^N B = f^{-1}(A)$.

Theorem 3.3.7. *(Synthetic Myhill's theorem) In $\mathcal{E}ff$, a set A is creative if and only if A is complete.*

Proof. Suppose A is creative with u as in definition 3.3.2 and fix $B{:}\Sigma^N$. Construct $R{:}\Sigma^{(N^3)}$ by $\{ (x,y,z) \mid y{:}B \wedge u(x) = z \}$. Apply the Second recursion theorem 3.2.3 to get $n{:}N^N$ such that $\forall x : N\ W_{n(x)} = \{ z \mid x \in B \wedge u(n(x)) = z \}$. We claim that $f \equiv u \circ n$ is total. To this end apply (i) to $e = n(x)$ for a fixed $x : N$. Then either $\exists z : W_{n(x)} \cap A$, and that z has to be $u(n(x)) : N$ or $u(n(x)) : N$ outright. That proves our claim. Now suppose $x : B$, we know that $u(n(x)) : N$ so that the definition gives $u(n(x)) : W_{n(x)}$. By (ii) we have $u(n(x)) {:} W_{n(x)} \to \exists z {:} W_{n(x)} \cap A$ means $z = u(n(x)) = f(x){:}A$. On the other hand, suppose $f(x){:}A$ and apply $u(n(x)){:}A \to \exists z{:}W_{n(x)} \cap A$, but $z{:}W_{n(x)}$ only if $x{:}B$. Deduce $B = f^{-1}(A)$ as desired.

Conversely, suppose A is complete. Given a $B{:}\Sigma^N$ and an $f{:}N^N$ as in definition 3.3.6, we claim that A is creative whenever B is. Consider the following diagram:

$$\begin{tikzcd}
N \arrow[r, two heads, "W"] \arrow[d, dashed, "g"'] & \Sigma^N \arrow[d, "\Sigma f"] \\
N \arrow[r, two heads, "W"'] & \Sigma^N
\end{tikzcd}$$

We assert that it commutes. The upper right corners reads

$$\forall e{:}N \exists e'{:}N W_{e'} = f^{-1}(W_e)$$

and thus by ACC applied to N

$$\exists g{:}N^N \forall e{:}N W_{g(e)} = f^{-1}(W_e).$$

Now suppose B is creative with a creative morphism v as in definition 3.3.2. We assert that $u \equiv f_\perp \circ v \circ g$ is a creative morphism for A. Pick $r{:}N$ and look at $g(r)$. From (i) in definition 3.3.2 for B it follows that

$$\exists n{:}W_{g(r)} \cap B \quad \vee \quad v(g(r)){:}N;$$

by completeness of A and the above argument

$$\exists n{:}f^{-1}(W_r) \cap f^{-1}(A) \quad \vee \quad v(g(r)){:}N;$$

or equivalently

$$\exists n{:}f^{-1}(W_r \cap A) \quad \vee \quad v(g(r)){:}N.$$

Thus $f_\perp(n) : W_r \cap A$ or $f_\perp(v(g(r))) : N$, establishing (i) of definition 3.3.2 for A. Now suppose $u(r) : W_r \cup A$. Then $v(g(r)) : f^{-1}(W_r \cup A)$ or equivalently $v(g(r)) : W_{g(r)} \cup B$. Applying (ii) for B gives that $\exists n{:}W_{g(r)} \cap B$ and repeating the same argument establishes $\exists n{:}W_r \cap A$. This proves our claim. Now in particular, for $K{:}\Sigma^N$ there exists $f{:}N^N$ such that $K = f^{-1}(A)$ and the desired result follows from proposition 3.3.5. ∎

3.4 Conclusion and future work

The *s-m-n* theorem while being an important result in the classical world, has turned out to be a simple application of the axiom of countable choice in the effective topos. This structure was previously not present in the classical informal or formal proof. We showed that K being creative is a straightforward fact, despite the fact that our definition of creativeness is constructively stronger. Indeed, we have demonstrated that non-trivial facts about computability theory find their home in the effective topos. Synthetic Myhill's theorem is such an example. In general, the synthetic results made no explicit reference to Gödel encoding or Turing machines. As Andrej Bauer puts it "we just [did] ordinary math–in an extraordinary universe" [1].

There are various directions we can explore from here. One thing we want to investigate next is how Roger's admissable numbering would look like in the effective topos. Originally, the project started with looking at a paper by Moschovakis [17], where Myhill's theorem appeared among the applications of the Second recursion theorem. Another interesting result there concering partial recursive functionals is the Kreisel-Lacombe-Shoenfield-Ceiten theorem. As we did not use the full force of the effective topos, such a translation would show how higher-order computability results appear here.

Bibliography

[1] A. Bauer. “First Steps in Synthetic Computability Theory”. In: *Electronic Notes in Theoretical Computer Science* 155 (2006). Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI), pp. 5–31. ISSN: 1571-0661. DOI: `https://doi.org/10.1016/j.entcs.2005.11.049`. URL: `https://www.sciencedirect.com/science/article/pii/S1571066106001861`.

[2] A. Bucalo, C. Führmann, and A. Simpson. “Equational lifting monads”. In: *CTCS ’99: Conference on Category Theory and Computer Science (Edinburgh)*. Vol. 29. Electron. Notes Theor. Comput. Sci. Elsevier Sci. B. V., Amsterdam, 1999, Paper No. 29004, 32.

[3] D. van Dalen. *Logic and structure*. Fifth. Universitext. Springer, London, 2013, pp. x+263. ISBN: 978-1-4471-4557-8; 978-1-4471-4558-5. DOI: `10.1007/978-1-4471-4558-5`. URL: `https://doi.org/10.1007/978-1-4471-4558-5`.

[4] E. Ghiorzi. *Complete internal categories*. 2020. DOI: `10.48550/ARXIV.2004.08741`. URL: `https://arxiv.org/abs/2004.08741`.

[5] J. M. E. Hyland. “The effective topos”. In: *The L.E.J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*. Vol. 110. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam-New York, 1982, pp. 165–216.

[6] J. M. E. Hyland. “A small complete category”. In: *Ann. Pure Appl. Logic* 40.2 (1988), pp. 135–165. ISSN: 0168-0072. DOI: `10.1016/0168-0072(88)90018-8`. URL: `https://doi.org/10.1016/0168-0072(88)90018-8`.

[7] P. T. Johnstone and E. P. Robinson. “A note on inequivalence of realizability toposes”. In: *Math. Proc. Cambridge Philos. Soc.* 105.1 (1989), pp. 1–3. ISSN: 0305-0041. DOI: `10.1017/S0305004100001304`. URL: `https://doi.org/10.1017/S0305004100001304`.

[8] S. C. Kleene. *Introduction to metamathematics*. D. Van Nostrand Co., Inc., New York, N. Y., 1952, pp. x+550.

[9] S. C. Kleene. “Realizability: a retrospective survey”. In: *Cambridge Summer School in Mathematical Logic (Cambridge, 1971)*. 1973, 95–112. Lecture Notes in Math., Vol. 337.

[10] J. Lambek. “Functional completeness of cartesian categories”. In: *Ann. Math. Logic* 6 (1973/74), pp. 259–292. ISSN: 0003-4843. DOI: 10.1016/0003-4843(74)90003-5. URL: https://doi.org/10.1016/0003-4843(74)90003-5.

[11] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*. Vol. 7. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 1986, pp. x+293. ISBN: 0-521-24665-2.

[12] F. W. Lawvere, ed. *Toposes, algebraic geometry and logic*. Lecture Notes in Mathematics, Vol. 274. Partial Report on a Conference on Connections between Category Theory and Algebraic Geometry & Intuitionistic Logic, Dalhousie University, Halifax, Nova Scotia, January 16–19, 1971. Springer-Verlag, Berlin-New York, 1972, pp. v+189.

[13] F. William Lawvere and Stephen H. Schanuel. *Conceptual mathematics*. Second. A first introduction to categories. Cambridge University Press, Cambridge, 2009, pp. xviii+390. ISBN: 978-0-521-71916-2. DOI: 10.1017/CBO9780511804199. URL: https://doi.org/10.1017/CBO9780511804199.

[14] T. Leinster. *Basic category theory*. Vol. 143. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2014, pp. viii+183. ISBN: 978-1-107-04424-1. DOI: 10.1017/CBO9781107360068. URL: https://doi.org/10.1017/CBO9781107360068.

[15] John R Longley. “Realizability toposes and language semantics”. In: (1995).

[16] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic*. Universitext. A first introduction to topos theory, Corrected reprint of the 1992 edition. Springer-Verlag, New York, 1994, pp. xii+629. ISBN: 0-387-97710-4.

[17] Y. N. Moschovakis. “Kleene’s amazing second recursion theorem”. In: *Bull. Symbolic Logic* 16.2 (2010), pp. 189–239. ISSN: 1079-8986. DOI: 10.2178/bsl/1286889124. URL: https://doi.org/10.2178/bsl/1286889124.

[18] J. van Oosten. *Realizability: an introduction to its categorical side*. Vol. 152. Studies in Logic and the Foundations of Mathematics. Elsevier B. V., Amsterdam, 2008, pp. xvi+310. ISBN: 978-0-444-51584-1.

[19] H. Rogers Jr. *Theory of recursive functions and effective computability*. Second. MIT Press, Cambridge, MA, 1987, pp. xxii+482. ISBN: 0-262-68052-1.

[20] M. Schönfinkel. “Über die Bausteine der mathematischen Logik”. In: *Math. Ann.* 92.3-4 (1924), pp. 305–316. ISSN: 0025-5831. DOI: 10.1007/BF01448013. URL: https://doi.org/10.1007/BF01448013.

[21] D. Scott. “Extending the topological interpretation to intuitionistic analysis”. In: *Compositio Math.* 20 (1968), 194–210 (1968). ISSN: 0010-437X.

[22] R. I. Soare. *Turing computability*. Theory and Applications of Computability. Theory and applications. Springer-Verlag, Berlin, 2016, pp. xxxvi+263. ISBN: 978-3-642-31932-7; 978-3-642-31933-4. DOI: 10.1007/978-3-642-31933-4. URL: https://doi.org/10.1007/978-3-642-31933-4.