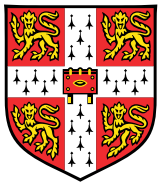


University of Cambridge

Mathematical Tripos

Computability and Logic

Lecturer:
Dr. T.E. Forster

Editor:
H-J. Wagenaar

Contents

Chapter 1

Recursion

1.1 Character

Definition. A **recursive datatype**, also **inductively defined set**, has founders and constructors. It is of

- **Finite character** if the constructors have finite arity.
- **Bounded character** if the constructors have bounded arity.
- **Unbounded character** (absolutely infinite) if constructors have unbounded arity.

Examples. The following have finite character.

(i) $\mathbb{N}$ has a recursive datatype declaration in the form of founders (0) and constructors (S, the successor function).

(ii) HF has founder $\emptyset$ and if $x, y \in \text{HF}$ then $x \cup \{y\} \in \text{HF}$.

(iii) α-lists are generated by the empty-list [] and if $x \in \alpha$, l is an α-list, then $x :: l$ is an α-list.

The following have bounded characters, by ω, constructors.

(iv) HC has founder $\emptyset$ and constructor countable union.

(v) Borel sets have founders the open sets, and the constructors are countable unions and the complement.

The following has unbounded character.

(vi) The class of ordinals: Ord.

1.2 Fix Point

$$\mathbb{N} = \bigcap \{X : 0 \in X \cap S``X \subseteq X\}$$
$$\frac{F(0) \quad (\forall n)(F(n) \to F(Sn))}{(\forall n \in \mathbb{N})(F(n))}$$

This induction works for $\mathbb{N} \subseteq \{n : F(n)\}$ as required. Every inductively defined set is the least fixed point for some nice function $V \to V$, for $\mathbb{N}$ it is $x \mapsto x \cup \{0\} \cup S``x$. For example, the transitive closure of a relation R:

$$t(R) = \bigcup_{n \in \mathbb{N}} R^n = \bigcap \{S \supseteq R : S \circ S \subseteq S\}.$$

Transitivity, symmetry and totality are given by

$$(\forall x, y, z)(R(x, y) \wedge R(y, z) \to R(x, z)),$$
$$(\forall x, y)(R(x, y) \to R(y, x)),$$
$$(\forall x, y)(R(x, y) \vee R(y, x)).$$

This is also do-able for transitivity and symmetry, as they are of the form:

$$\bigwedge_{i \in I} r_i(\vec{x}) \to S(\vec{x}).$$

Definition. **Horn clauses** are disjunction of (negations of) atomics of which at most one disjunct is not a negation.

A **Horn theory** has axioms universal closures of Horn clauses.

An **algebraic theory** has axioms universal closures of equations.

Note. A theory has universal axioms if and only if every substructure of any model is another model of such theory.

An inductively defined set is of the form $\subseteq$-least set containing X and satisfying F. If F is Horn, such a set is guaranteed to exist.

T is an algebraic theory if and only if the class of its models is closed under arbitrary products, substructure and homomorphisms (Birkhoff).

Groups, rings and integral domains are Horn theories.

1.3 Structural Induction

Definition. Recursive datatypes have an **ENGENDERING RELATION**, defined by its constructors.

Recursive data types support **STRUCTURAL INDUCTION**, which is induction on the engendering relation. Given a relation R (of arity 2, say) it is called R-**INDUCTION**:

$$\frac{(\forall x)((\forall y)(R(y,x) \to F(y)) \to F(x))}{(\forall x)(F(x))}$$

Inductively defined sets have **CERTIFICATES** for its members.

A recursive data type is **FREE** if all elements have only one certificate.

If a recursive data type is not free and not of finite character AC is needed to solve two problems: uniqueness and existence. For $\mathbb{N}$ and the language of propositional logic, Infinity, Replacement and then Separation gives existence and uniqueness follows by taking intersection. HC and HF are obtained similarly as the ωth and $\aleph_1$th stage of using $\mathcal{P}$ limited to subsets of size $\aleph_0$.

Exercise 1.3.1 (TRIPOS 2006, Q. 12, QUINE'S TRICK). Let $P(|x|)$ be $|x \backslash \{y\}|$ if $y \in X$ and 0 if X is empty. Define

$$q(n) \iff (\forall Y)((n \in Y \wedge (P\text{“}Y \subseteq Y)) \to 0 \in Y).$$

Establish that $q(n)$ if and only if n is a natural number.

Theorem 1.1. The engendering relation on a recursive datatype is well-founded.

Proof. Suppose not: $(\exists x)(\neg F(x)) \wedge (\forall x)((\forall y)(R(y,x) \to F(y)) \to F(x))$. Set $\{x \colon \neg Fx\} = A$ (foregoing set-theoretical difficulties at this point):

$$(\forall x \in A)(\exists y \in A)(R(y,x)) \wedge (A \neq \emptyset).$$

This has negation:

$$(\forall A \subseteq \mathrm{dom}(R))[A \neq \emptyset \to (\exists x \in A)(\forall y \in A)(\neg R(y,x))]. \qquad \square$$

1.4 Restricted Quantifiers

Definition. **Restricted** or **guarded quantifiers** are for example $(\forall x \in y)$ which seems equivalent to $(\forall x)(x \in y \Rightarrow \cdots)$.

The **Prenex Normal Form** is any sentence of the form a number of quantifiers and then no quantifiers at all.

Theorem 1.2 (PNF Theorem). Every formula is equivalent to one in which all unrestricted quantifiers lie outside the restricted quantifiers.

Proof. The result inductively rests on **quantifier pushing** by replacing

$$(\forall x \in A)(\exists \varphi)$$

with

$$(\exists B)(\forall x \in A)(\exists y \in B)\varphi$$

where B exists by the Collection. Although we have introduced another quantifier, it is restricted and on the right side. □

1.5 Infinitary horn

Theorem 1.3. There is no first-order theory of well-founded relation.

Proof. A simple compactness argument. □

Let $L_{\kappa\lambda}$ be a language allowing $< \kappa$ conjunctions or disjunctions and allowing to bind $< \lambda$ variables with one block of $\forall$ or $\exists$. The normal language is $L_{\omega\omega}$. $L_{\omega_1\omega}$ is important, and $L_{\omega_1\omega_1}$ allows one to express well-foundedness:

$$\forall x_1, \ldots, x_n(\neg R(x_2x_1) \vee \neg R(x_3x_2) \vee R(x_3x_1) \vee \cdots)$$

In full ZF we have that (powersets, $\in$) are well-founded, yet are not a recursive datatype.

Chapter 2

Functions

2.1 Primitive Recursion

Definition. A FUNCTION-IN-INTENSION is the algorithm/program.

A FUNCTION-IN-EXTENSION is the graph of a function.

f"X is the RANGE of f under X

Definition. A PRIMITIVE RECURSIVE FUNCTION is in extension $\mathbb{N}^k \to \mathbb{N}$, constructors are:

(i) $\lambda n.0$, the identically zero function.

(ii) Projections functions: giving back ith member of tuple.

(iii) Composition.

(iv) Primitive recursion, if g and r are primitive recursive, then so is

$$f(0, \overline{x}) = g(\overline{x})$$
$$f(Sn, \overline{x}) = h(f(n, \overline{x}), n, \overline{x}).$$

Examples. (i) Predecessor function: $P(0) := 0$ and $P(S(x)) = x$.

(ii) $x \dot{-} 0 = x$ and $x \dot{-} S(y) = P(x \dot{-} y)$.

(iii) $x \cdot 0 = 0$ and $x \cdot Sy = x \cdot y + x$.

(iv) $x \wedge 0 = S(0)$ and $x \wedge sy = (x \wedge y) \cdot x$.

(v) `if-then-else`: $(0, x, y) = x$ and $(S(n), x, y) = y$.

These are part of the Doner-Tarski Hierarchy.

(v) $\sum_f(n) := \sum_{0 \leqslant x \leqslant n} f(x)$ where f is primitive recursive.

(vi) $\prod_f(n) := \prod_{0 \leqslant x \leqslant n} f(x)$ where f is primitive recursive.

One solves circularity by using the function in extension: calculation: $y = f(x)$ by a certificate giving the input and outputs for inputs $x' \leqslant x$.

(i) $x \mid y$ if $x = y \vee (\exists w < y)(x \cdot w = y)$,

(ii) $x \,\mathtt{div}\, y$ is the largest z such that $y \cdot z \leqslant x$,

(iii) $x \,\mathtt{rem}\, y$ is the remainder when x is divided by y,

(iv) p is a prime $(\forall x < p)(\forall y < p)(x \cdot y \neq p)$,

(v) z is a power of p if $(\forall w < z)(w \mid z \to p \mid w)$.

This is in the language of ordered rings. One cannot recover the exponent. We can encode the certificate by using a big enough prime and using

$$O_z = (O \,\mathtt{rem}\, z) \,\mathtt{div}\, (z/p)$$

to extract it again. Now the statement that there is a certificate for $f(x, \overline{s}) = y$ is given by $(\exists I)(\exists O)(\exists p)$ and for any $z < I$ that is a power of p,

(i) $\langle I_z, O_z \rangle$ is related by recursion for f, $O_z = h(O_{z/p}, I_{z/p}, \overline{s})$,

(ii) $I_1 = 0$; $O_1 = g(0, \overline{s})$,

(iii) $I_z = I_{z/p} + 1$

(iv) and there exists z such that $x = I_z$ and $y = O_z$.

The current statement has $(\forall z < I)(\exists m)(\varphi(z, m))$ which can be quantifier-pushed to $(\exists w)(\forall z < I)(\exists m < w)(\varphi(z, m))$. The four unrestricted quantifiers which are then at the beginning can then be grouped together by using one unrestricted quantifier as an upper bounded for the quadruple.

One can encode finite sequences by using prime numbers and exponentiation.

Primtive Recursive Relations

Definition. $R(\overline{x})$ is a **PRIMITIVE RECURSIVE RELATION** if and only if there exists a primitive recursive function r such that $r(\overline{x}) = 0$ if and only if $R(\overline{x})$.

Examples. (i) $<_{\mathbb{N}}$, $\leqslant_{\mathbb{N}}$ and $=$ are primitive recursive.

(ii) R is primitive recursive, then $\neg R$ is primitive recursive.

(iii) R and S primitive recursive, then so is $R \cap S$ and $R \cup S$.

(iv) R^{-1} is where R is binary, and in general for permutations.

(v) Not composition.

(vi) Closed under substitution.

(vii) Restricted quantification.

If $R(x, \overline{y})$ is represented by $r(x, \overline{y})$ then $(\exists x \leqslant z)(R(x, \overline{y}))$ represented by

$$\prod_{0 \leqslant x \leqslant z} r(x, \overline{y}).$$

2.2 μ-induction

μ-minimisation is given by $(\mu x)(gx = y)$. In the general machines later, it will only be allowed once.

Ackermann Function

Simultaneous induction does not add much. Consider

$$\begin{aligned} f(0, \overline{x}) &= g(\overline{x}) \\ f(S(n), \overline{x}) &= g(f(n, \overline{x}), n, \overline{x}). \end{aligned}$$

Particularly

$$\begin{aligned} A(0, n) &= S(0) \\ A(S(m), 0) &= A(m, 1) \\ A(S(m), S(n)) &= A(m, A(S(m), n)) \end{aligned}$$

A is a total function by double induction or well-founded induction.

Definition. $f \colon \mathbb{N} \to \mathbb{N}$ **DOMINATES** $g \colon \mathbb{N} \to \mathbb{N}$ if there exists n such that for all $m > n$, $f(m) > g(m)$.

For every primitive recursive function $f \colon \mathbb{N}^k \to \mathbb{N}$ there is a constant c_f such that for all $\overline{x}$, $f(\overline{x}) < A(c_f, \max(\overline{x}))$. So A cannot be primitive recursive.

Chapter 3

Machines

3.1 Finite State Machines

Definition. **Finite state machine** (**FSM** or **DFA**) consists of finitely many states, a transition table and a list of **accepted** states.

Let $L \subseteq \Sigma^*$, which is $\Sigma^{<\omega}$, finite strings with characters from a finite set, M **recognizes** L if $L = \{w \in \Sigma^* \colon M \text{ accepts } w\}$.

$L \subseteq \Sigma^*$ is **regular** if there exists M such that M recognises L.

Regular Expressions are defined recursively. Any letter from Σ is a regular expression. if X, Y are, so are XY and $X \mid Y$ $(= X \cup Y)$ and X^*.

Automatic structures are those computable by DFA: **automatic groups** have multiplication table computed by a DFA.

Examples. (i) Boolean combinations of regular languages are regular.

(ii) $L_1L_2 = \{w_1w_2 \colon w_1 \in L_1,\ w_2 \in L_2\}$ is regular if L_1 and L_2 are.

(iii) L^* (Kleene star) if L is, closure under concatenation.

(iv) L reversed is regular if L is.

Theorem 3.1 (Kleene's Theorem). A language is regular if and only if it is denoted by a regular expression.

Proof sketch. $\Longleftarrow$ is trivial. The inductive step of the result is done on the size of Q, a subset of the states.

For any two states q_1, q_2 and any subset Q of the sets of states, there is a regular expression that captures the set of strings taking the machine from q_1 to q_2 without leaving Q. □

Lemma 3.2 (Pumping Lemma). Given M accepts w, and $|M| < |w|$, then a state is at least visited twice. Split $w = w_0 w_1 w_2$ such that w_1 is the inside of a repeat. Then $w_0 (w_1)^n w_2$ is also accepted.

Theorem 3.3 (Myhill-Nerode Theorem). A language L is regular if and only if $\sim_\infty$ has finite index.

Proof. Consider w_1 and w_2 equivalent if there is no difference in the language; $w_1 \sim_\infty$ if $w_1 \sim_n w_2$ for all n (intersection):

$$\begin{aligned} w_1 \sim_0 \quad w_2 &\iff (w_1 \in L \iff w_2 \in L) \\ w_1 \sim_{n+1} w_2 &\iff (\forall c \in \Sigma)(w_1 :: c \sim_n w_2 :: c) \wedge w_1 \sim_n w_2. \end{aligned}$$

The quotient under $\sim_\infty$ is a machine: a state is an equivalence of words, unique minimal machine. □

Non-Deterministic Machines

NFA are **non-deterministic finite automaton**, it accepts w if it "might" accept it, if the machine guesses "right". An NFA is useful in proving that concatenation preserves regularity. For every NFA there is a canonical DFA that recognizes the same language: take the power set of states and set transition table and acceptance states similarly.

PDA are **Push-down Automaton**, also known as **context-free languages**. For example the language given by $\Sigma = \{(,)\}$ of correctly opened and closed braces. This cannot be captured by a regular expression: it needs a **stack** on which has a pop and push, and is originally empty. Deterministic and non-deterministic variants are *not* equivalent.

Recursive Ordinals

A **RECURSIVE ORDINAL** is the order type of a well-ordering of the natural numbers whose graph is a decidable set. α^β, given $\langle \mathbb{N}, <_\alpha \rangle$ and $\langle \mathbb{N}, <_\beta \rangle$ is the order type of $f\colon \langle \mathbb{N}, <_\alpha \rangle \to \langle \mathbb{N}, <_\beta \rangle$ of finite support (all but finitely many values are 0_A—bottom element in the sense of $<_A$) ordered lexicographically.

The set of recursive ordinals is closed under taking initial segments. There are countably many computable ordinals, and there are uncountable many ordinals. Hence ω_1^{CK} exists, the least ordinal that is not recursive, and it is countable. Every countable ordinal has cofinality $< \omega$, and each recursive ordinal has a computable cofinal sequence of length ω.

3.2 General Machines

Turing Machines. Operates on a tape, can move left, right, write, erase or halt according to the symbol (normally 0 or 1) at the current tape position and state using a table.

Register Machines. Finitely many registers holding natural numbers and a program with a finite list of instructions. Each instruction has a label and a body. Labels are natural numbers, and a body is

- Add 1 to the contents of R and jump to instruction with label L.
- If content of R is non-zero, subtract 1 and jump to the instruction with label L', otherwise jump to instruction with label L'.
- Halt.

Definition. The function-in-extension of such a machine is called **PARTIAL COMPUTABLE** or **PARTIAL/GENERAL RECURSIVE**.

There is a **GÖDEL NUMBERING** of machines by natural numbers.

(i) $\{e\}(n)\!\downarrow = k$ if the machine coded by e halts with input n and output k.

(ii) $\{e\}(n)\!\uparrow$ if the machine coded by e does not halt with input n, otherwise known as **DIVERGES**.

(iii) $\{e\}_x(n)\!\downarrow$ halts in at most x steps.

(iv) $\{e\}_x(n)\!\downarrow = z$ halts in at most x steps at z.

(v) $W_e = \{n \in \mathbb{N}\colon \{e\}(n)\!\downarrow\}$.

Universal Machine

The μ-recursive functions are precisely those computed by register machines. The key gadget is Kleene's T-function:

$T(p,i,t) =$ "complete course of computability" of machine with number p run on input i for t steps.

$T(p,i,t)$ is a list of core-dumps for times $0 \leqslant t' \leqslant t$ and it is primitive recursive. So there is a machine that computes it! Suppose d is a dump:

$$\texttt{current\_instruction}(d)$$
$$\texttt{register}_0(d)$$
$$\texttt{last}(l) = \text{ last element of the list } l.$$

There is a machine on given m and i diverges or outputs as $\{m\}(i)$:

$$\texttt{register}_0(\texttt{list}(T(m,i,(\mu t)(\texttt{current\_instruction}(\texttt{list}(T(m,i,t))) = \texttt{halt}))))$$

3.3 Decidable and Semi-Decidable Sets

Definition. $A \subseteq \mathbb{N}$ is **SEMI-DECIDABLE** if A is $\{n\}$" for some n.

Also called **FINITELY RECOGNISABLE**, **COMPUTABLY ENUMERABLE** or **RECURSIVELY ENUMERABLE**.

A is **DECIDABLE** if A and $\mathbb{N}\backslash A$ are both semi-decidable.

A **VOLCANO** is a machine that runs $T(m,i,t)$ by a bijection $\mathbb{N} \to \mathbb{N}^2$ and outputs whenever $\{m\}$ has actually computed some output.

Theorem 3.4. For $\emptyset \subsetneq A \subseteq \mathbb{N}$ the following are equivalent:

(i) $A = \{n\}$" for some n,

(ii) $A = \{n\}$" for some n and $\{n\}$ is total,

(iii) $A = W_n$ for some n.

Proof. **(i) $\implies$ (ii).** Take $\{n\}$ with $\{n\}" = A$ and put it in a volcano. Now we wrap this in a machine that on input i outputs the output of the volcano for the ith step on traversing $\mathbb{N}^2$, if there is no output there, it outputs the first output, this is $\{n'\}$. As A is non-

empty, we will know the volcano will output some number, so $\{n'\}$ is total, and as the volcano traverse $\mathbb{N}^2$, we have $\{n'\}\text{“} = A$. □

(ii) $\Longrightarrow$ (i). This is clear. □

(i) $\Longrightarrow$ (iii). Take $\{n\}$ with $\{n\}\text{“} = A$ and put it in a volcano. Now wrap this in a machine that on input i halts (and outputs anything) when it detects i as an output in the volcano. This halts on input i if and only if $i \in \{n\}\text{“}$, so have n' with $W_{n'} = A$. □

(iii) $\Longrightarrow$ (i). Say $A = W_n$, which we put in a volcano and then again, traversing $\mathbb{N}^2$ to ith step, if $\{n\}$ has halted by these numbers of step on this input outputs the input, otherwise it loops. □

Theorem 3.5. Let $X \subseteq \mathbb{N}^{k-1}$. Then X is the projection of a decidable subset of $\mathbb{N}^k$ if and only if it is semi-decidable

Proof. $\Longrightarrow$ Suppose X is $\{\underline{x}\colon (\exists n)(\underline{x} :: n \in Y)\}$ where $Y \subseteq \mathbb{N}^k$ is decidable.

For a candidate tuple $\underline{x} \in \mathbb{N}^{k-1}$, it can be determined whether $\underline{x} \in X$ by testing $\underline{x} :: 0$, $\underline{x} :: 1$, ... for membership of Y. □

$\Longleftarrow$ Suppose $X = \mathrm{dom}(f)$ for some computable $\{n\}$. $\underline{x} \in X$ if and only if $\{m\}(\underline{x})\downarrow$ if and only if $(\exists y)(\{m\}_y(\underline{x})\downarrow)$ if and only if $(\exists y)(\langle \underline{x}, y\rangle \in \{\langle \underline{z}, y\rangle\colon \{m\}_y(\underline{z})\downarrow\})$. □

Note. X is decidable if and only if its characteristic function is.

Immune Sets

Definition. An infinite $X \subseteq \mathbb{N}$ is IMMUNE it has no infinite semi-decidable subset.

A natural example is given by the following idea. Consider functions with output finite strings of 0s and 1s. There is a universal Turing machine U which for any computable f will compute $f(\tau)$ as follows: U will associate to each such f a string ρ_f such that for any τ, $f(\tau)$ is obtained as $U(\rho_f :: \tau)$. We say $C(\sigma)$ is the minimal $|\rho_f :: \tau'|$ where $f(\tau') = \tau$.

Remark. $\{\sigma \in \{0,1\}^{<\omega} : C(\sigma) \geqslant |\sigma|/2\}$ is immune.

Proof. Suppose it has an infinite semi-decidable subset $B = f\text{“}$. $|B| = \aleph_0$, so B contains strings of arbitrary length. Let h_n be the first string of length $\geqslant n$ that f puts into B.

By assumption $C(h_n) \geqslant |h_n|/2 \geqslant n/2$. Manifestly h_n can be obtained from n by computing! So this gives $C(h_n) = C(n) + |\rho_f|$ where ρ_f is the string that U uses to compute f', some appropriate modification of f. However, $C(n) \leqslant \log_2(n)$. So gives $n/2 \leqslant \log_2(n) + |\rho_f|$ which for large n does not hold. □

Semi-decidable sets in V

Consider $f : V \to V$ a set-like function. Then $f(x)$ is always a set, whereas $f\text{“}x$ is only a set when we have Replacement, so not in Z! Similarly we can have $f\text{“}x$ a set, but $f\text{“}x \neq \{f\text{“}y : y \in x\}$. Computable functions are set-like.

Applications to Logic

Definition. A theory is a deductively closed set of formulae is **AXIOMATIZABLE** if and only there is a semi-decidable set of axioms of which it is the deductive closure.

The set of theorems is a projection of the set of proofs. So a decidable set of axioms leads to a semi-decidable set of theorems, but so do semi-decidable sets! Fortunately we have the following.

Theorem 3.6 (Craig's Theorem). Every theory with a semi-decidable set of axioms has a decidable set of axioms.

3.4 The Halting Problem

Theorem 3.7. $\{\langle p, i\rangle : \{p\}(i)\downarrow\}$ is not decidable.

Proof. Suppose $\mathcal{M}$ such that if the input is n, code for p and i: if $\{p\}(i)\downarrow$ then "yes", if $\{p\}(i)\uparrow$ then "no".

Modify $\mathcal{M}$ by trapping the output so halt if $\mathcal{M}(n)\downarrow$ is no, then we halt and say "no". If "yes", then loop forever. Call this machine $\mathcal{M}^*$.

Modify this machine by putting a front end that accepts input x and feeds the pair $\langle x, x\rangle$ to $\mathcal{M}^*$. Call this machine $\mathcal{M}^{**}$. It has gnumber m. What happens when we feed $\mathcal{M}^{**}$ the number m?

Then in extension we get $\mathcal{M}^*(m, m)$ and $\mathcal{M}^{**}(m)$ halts if and only if it does not halt: $h \leftrightarrow \neg h$ □

Not the same as $h \wedge \neg h$: we do *not* need the law of the excluded middle.

Rice's Theorem

Theorem 3.8 (The $S - m - n$ Theorem). There is a computable total function S such that for all e, b, $a \in \mathbb{N}$ we have$\{e\}(b, a) = \{S(e, b)\}(a)$.

Corollary 3.9 (Fixed Point Theorem). Let $h\colon \mathbb{N} \to \mathbb{N}$ be total computable. Then $\exists n \in \mathbb{N}$ with $\{n\} = \{h(n)\}$.

Proof. Consider the map $\text{pair}(e, x) = \{h(S(e, e))\}(x)$. This is a computable function, so is computable by a machine with gnumber a. Set $n := S(a, a)$, now

$$\{n\}(x) = \{S(a, a)\}(x) = \{a\}(a, x) = \{h(S(a, a))\}(x) = \{h(n)\}(x). \quad \square$$

Theorem 3.10 (Rice's Theorem). Let $\emptyset \subsetneq A \subseteq \mathbb{N}$ then $\{n\colon \text{Graph}(\{n\}) \in A\}$ is not decidable.

Proof. Suppose it is: χ_A is computably total. Find a, $b \in \mathbb{N}$ such that $\text{Graph}(\{a\}) \in A$ and $\text{Graph}(\{b\}) \notin A$. Obtain computable g such that:

$$g(n) = \texttt{if } \text{Graph}(\{n\}) \in A \texttt{ then } b \texttt{ else } a.$$

Now by the Fixed Point Theorem, there exists n with $\{n\} = \{g(n)\}$.

Suppose $\text{Graph}(\{n\}) \in A$ then $\text{Graph}(\{g(n)\}) \in A$. So $g(n) = b$, $g(g(n)) = b$ and $g(b) = b$.

Suppose $\text{Graph}(\{n\}) \notin A$ then $\text{Graph}(\{n\}) \notin A$. So $g(n) = a$, $g(g(n)) = a$ and $g(a) = a$. □

Again we do not need the law of the excluded middle to reach the conclusion.

3.5 Recursive Inseparability

Definition. Two disjoint sets X, Y are said to be **RECURSIVE INSEPARABLE** if there is no decidable set Z with $X \subseteq Z$ and $Y \cap Z = \emptyset$.

Proposition 3.11. The two sets

$$A = \{e : \{e\}(e)\downarrow > 0\}$$
$$B = \{e : \{e\}(e)\downarrow = 0\}$$

are recursively inseparable.

Proof. Suppose $f : \mathbb{N} \to \{0, 1\}$ is a total function for which $f``A = \{0\}$ and $f``B = \{1\}$ then f is not computable. Consider $n \in \mathbb{N}$. We will show that $\{n\}$ is not an f as above.

$\{n\}(n)\uparrow$. $\{n\} \neq f$ because f is total.

$\{n\}(n)\downarrow$. $\{n\}(n)\downarrow = 0$. Then $n \in B$ and thus $\{n\}(n) \neq f(n)$.

$\{n\}(n)\downarrow > 0$. Then $n \in A$ and thus $\{n\}(n) \neq f(n)$. □

Chapter 4

λ-Calculus

In λ-calculus there are the following conversion rules:

α-conversion. $\lambda x.f \to_\alpha \lambda y.f[y/x]$.

β-conversion. $(\lambda x.f)g \to_\beta f[g/x]$.

η-conversion. $\lambda x.fx \to_\eta f$.

This leads to an equivalence relation of functions-in-extension. This allows

$$r := \lambda x.\,\mathtt{not}(xx)$$

and we have

$$rr = (\lambda x.\,\mathtt{not}(xx))(\lambda x.\,\mathtt{not}(xx)) \to_\beta \mathtt{not}(rr) \to_\beta \mathtt{not}\,\mathtt{not}(rr)$$

which is **Russell's Paradox**.

4.1 Partial Computable Functions as λ-Terms

Claim 4.1. For every μ-recursive function $\mathbb{N}^k \to \mathbb{N}$ there is a λ-term which computes that function on Church numerals.

Booleans are defined thus:

$$\begin{aligned}\mathtt{true} &:= \lambda xy.x\\ \mathtt{false} &:= \lambda xy.y.\end{aligned}$$

Both are of type $A \to (B \to C)$, they are $A \to (B \to A)$ and $A \to (B \to B)$ and they are the only terms of this type. We have conditionals

$$\begin{aligned} \texttt{if-then-else} &:= \lambda bxy.bxy \\ \texttt{iszero} &:= \lambda n.n(\lambda x.\,\texttt{false})\,\texttt{true}\,. \end{aligned}$$

Pairs are defined thus:

$$\begin{aligned} \texttt{pair} &:= \lambda xyf.fxy \\ \texttt{fst} &:= \lambda p.p\,\texttt{true} \\ \texttt{snd} &:= \lambda p.p\,\texttt{false} \\ \texttt{nil} &:= \lambda x.\,\texttt{true}\,. \end{aligned}$$

Church numerals define the natural numbers:

$$\begin{aligned} \texttt{zero} &:= \lambda fx.x \\ \texttt{once} &:= \lambda fx.fx \\ \texttt{twice} &:= \lambda fx.f(fx) \\ \texttt{thrice} &:= \lambda fx.f(f(f(x))). \end{aligned}$$

For `plus` we use $A \times B \to C$ equivalent to $A \to (B \to C)$:

$$\begin{aligned} \texttt{succ} &:= \lambda nfx.f((nf)x) \\ \texttt{plus}(n,m) &:= \lambda nmfx.(mf)((nf)x) \\ \texttt{mult}(n,m) &:= \lambda nmfx.m(nf)x = \lambda nmf.m(nf). \\ \texttt{exp}(n,m) &:= \lambda nmfx.mnfx = \lambda nm.mn \end{aligned}$$

Lists are pairs of heads and tails which are a list, or empty. Test for null list:

$$\texttt{null} := \lambda p.p(\lambda xy.\,\texttt{false}).$$

Consider $f\colon \mathbb{N} \to \mathbb{N}$ defined by recursion on $\mathbb{N}$, for example

$$\begin{aligned} \texttt{fact}(n) := \texttt{if}\ & n = 0 \\ \texttt{then}\ & 1 \\ \texttt{else}\ & n \cdot \mathrm{fact}(n-1). \end{aligned}$$

`fact` is fixed point for `metafact` of the form $(\mathbb{N} \to \mathbb{N}) \to (\mathbb{N} \to \mathbb{N})$:

$$\begin{aligned} \texttt{metafact}(f,n) := \texttt{if}\ & n = 0 \\ \texttt{then}\ & 1 \\ \texttt{else}\ & n \cdot f(n-1). \end{aligned}$$

Suppose $f = \texttt{metafact}\, f$ then $f = \texttt{fact}(n)$. The following **FIX POINT COMBINATOR** satisfies $Yf = f(Yf)$ and Yf is a fix point of f:

$$Y := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)).$$

Consider $\texttt{map}(f, l)$, which gives a list of values of f applied to members of l:

$$\begin{aligned}\texttt{map}(f.l) := Y(\lambda mfl.&\texttt{if } (\texttt{null}\, l)\\ &\texttt{then nill}\\ &\texttt{else } f(\texttt{fst}(l)) :: (m.f(\texttt{snd}(l))))).\end{aligned}$$

Write $\mathbb{N}$ for the stream of naturals, a list with no end:

$$\mathbb{N} := Y(\lambda l.0 :: \texttt{map succ}\, l).$$

Suppose $x = \texttt{map}(f, \mathbb{N})$ then $\texttt{mu}(n, x)$ gives the first input which gives output n:

$$\begin{aligned}\texttt{mu}\, nx := Y(&\texttt{if } (\texttt{snd}(\texttt{hd}\, x) = n)\\ &\texttt{then fst}(\texttt{hd}\, x)\\ &\text{else } \texttt{mu}\, n(\texttt{tl}(x))).\end{aligned}$$

4.2 Curry-Howard Correspondence in Proofs

Definition. One can decorate some natural deduction in constructive proofs with λ-terms, leading to a **TYPED λ-CALCULUS**.

This is the **CURRY-HOWARD CORRESPONDENCE**.

Symbol	Introduction	Elimination
		$\dfrac{x_A\colon A \quad y_B\colon B}{x_A\colon A}$ and $\dfrac{x_A\colon A \quad y_B\colon B}{y_B\colon B}$
$\wedge$	$\dfrac{x_A : A \quad y_B : B}{\texttt{pair}(x_A, y_B)\colon A \wedge B}$	$\dfrac{x_{A\times B}\colon \quad A \wedge B}{\texttt{fst}(x_{A\times B})\colon A}$ and $\dfrac{x_{A\times B}\colon \quad A \wedge B}{\texttt{snd}(x_{A\times B})\colon B}$
$\vee$	$\dfrac{x_A\colon A}{\texttt{pair}(x_A, 0)\colon A \vee B}$ and $\dfrac{x_B\colon B}{\texttt{pair}(x_B, 1)\colon A \vee B}$	$\dfrac{\overset{[A]}{\vdots}\ C \qquad \overset{[B]}{\vdots}\ C \qquad A \vee B}{C}$
$\rightarrow$	$\dfrac{\overset{[x_A\colon A]}{\vdots}\ f_B\colon B}{\lambda x_A.f_B\colon A \to B}$	$\dfrac{f_A\colon A \quad x_{A\to B}\colon A \to B}{f_A x_{A\to B}\colon B}$
$\bot$		$\dfrac{\bot}{A}$

The law of the excluded middle ($\neg\neg A \to A$) is not in constructive logic and so for example $A \vee (A \to \bot)$ is unprovable and does not have a λ-term.

Disjunction with multiple variables can be formalized as before, then $\bot \implies A$ can be seen as that the empty disjunct is false and so implies everything.

$\vee$-elimination and $\to$-introduction are not a lego block and are a pain.

Examples. (i) A simple example is:

$$\dfrac{\dfrac{\dfrac{x_A\colon [A]^2 \quad f_{A\to B}\colon [A\to B]^1}{\lambda x_A\colon B}}{\lambda f_{A\to B} x_A\colon (A\to B)\to B}\to\text{-int (1)}}{\lambda x_A.(\lambda f_{A\to B}.f_{A\to B}x_A)\colon A\to((A\to B)\to B)}\to\text{-int (2)}$$

(ii) A slightly more complex example is:

$$\dfrac{\dfrac{\dfrac{\dfrac{x\colon [A]^1 \quad y\colon [A\to(B\to C)]^3}{yx\colon B\to C} \quad \dfrac{x\colon [A]^1 \quad z\colon [A\to B]^2}{zx\colon B}}{(yx)(zx)\colon C}}{\lambda x.(yx)(zx)\colon A\to C}\to\text{-int (1)}}{\lambda z.\lambda x.(yx)(zx)\colon (A\to B)\to(A\to C)}\to\text{-int (2)}}{\lambda y_{A\to(B\to C)}.\lambda z_{A\to B}.\lambda x_A.(yz)(zx)\colon (A\to(B\to C))\to((A\to B)\to(A\to C))}\to\text{-int (3)}$$

(iii) The following cute result is due to Turing:

$$\dfrac{\dfrac{\dfrac{f\colon [(A\to B)\to A]^1 \quad x\colon [A\to B]^2}{fx\colon A} \quad x\colon [A\to B]^2}{B}}{\lambda x.x(fx)\colon (A\to B)\to B}\to\text{-int (2)}}{\lambda f.\lambda x.x(fx)\colon ((A\to B)\to A)\to((A\to B)\to B)}\to\text{-int (1)}$$

Chapter 5

Tennenbaum's Theorem

By the Compactness Theorem there is a non-standard model of PA, some of which are countable. Without loss of generality the carrier set of such a model is $\mathbb{N}$. Can the structure $(\mathbb{N}, 0, +, \cdot, s)$ be computable?

Consider $n \in m$ if and only if the nth bit of m (m written in binary) is 1 and Peano Arithmetic can be seen as a fragment of second order arithmetic.

In any non-standard model, the non-standard elements are added on to the end of $\mathbb{N}$. Let $\mathcal{M}$ be a non-standard model of PA and $\mathcal{M}^*$ the corresponding model of second order arithmetic.

Lemma 5.1. In $\mathcal{M}^*$, every decidable subset of the standard part of $\mathcal{M}$ is $\in$-encoded by some $x \in \mathcal{M}$.

Proof. Let us write $x \oplus y$ for the logical or of x, y thought of as bit-strings. $\oplus$ is primitive recursive. Let P be any decidable predicate of $\mathbb{N}$. Now we define $f\colon \mathbb{N} \to \mathbb{N}$ as follows

$$\begin{aligned} f(0) &= 0 \\ f(n+1) &= \text{ if } P(n) \text{ then } f(n) \oplus 2^n \text{ else } f(n) \end{aligned}$$

We have that

$$\{n \in \mathcal{M}\colon n \text{ is standard and } P(n)\}$$

is $\in$-encoded by $f(m)$ for sufficiently large (non-standard) m. □

Theorem 5.2 (Tennenbaum's Theorem). PA has no non-standard model in which the graphs of $+$ and $\cdot$ are decidable.

Proof. Suppose $+, \cdot$ computable in a given non-standard model. Suppose there were $m \in \mathbb{N}$ such that in the non-standard model, m $\in$-encoded the halting set (or any undecidable set).

We showed earlier that there were pairs of recursively inseparable sets. Let $\mathcal{A}$, $\mathcal{B}$ be a pair of recursively inseparable sets.

$$\mathcal{A} = \{n \colon (\exists y)(A(n, y))\}$$
$$\mathcal{B} = \{n \colon (\exists x)(B(n, x))\}$$

where A, B are decidable. So the standard model believes

$$(\forall n < \underline{m})(\forall y < \underline{m})(\forall x < \underline{m})(\neg A(n, y) \vee \neg B(n, x))$$

for any numeral $\underline{m}$. This is universal and so must be true in every model of PA, in particular, in $\mathcal{M}$. So $\mathcal{M}$ believes the above for *all* standard m. So it must hold for some non-standard m as well. Let e be such a non-standard element:

$$\mathcal{M} \models (\forall n < e)(\forall y < e)(\forall x < e)(\neg A(n, y) \vee \neg B(n, x)). \tag{5.1}$$
$$X = \{n \in \mathbb{N} \colon \mathcal{M} \models (\exists y < e)(A(y, \underline{n}))\}.$$

Claim. X separates $\mathcal{A}$ and $\mathcal{B}$.

Proof. $\mathcal{A} \subseteq X$ because any member of $\mathcal{A}$ satisfies $A(y, \underline{n})$ for some genuine natural, and any such is $< e$.

$\mathcal{B} \cap X = \emptyset$ holds for similar reasons. Suppose $n \in B$. Then there is some m such that $B(n, m)$ whence $\mathcal{M} \models B(\underline{n}, \underline{m})$ and thus $m < e$. So $\mathcal{M} \models (\exists m < e)(B(n, \underline{m}))$. But then, by (5.1) above $n \notin X$. $\square$

We have just shown that every non-standard model (with carrier set $\mathbb{N}$) encodes at least one undecidable set—to wit X. Thus such a model cannot be recursive.

Chapter 6

Incompleteness

Theorem 6.1. $\{n\colon \{n\} \text{ total}\}$ is not semi-decidable.

Proof. Suppose it were semi-decidable. Let f be the total computable function whose values are precisely the numbers of machines that compute total functions. Now consider the function $g; = \lambda n.\{f(n)\}(n) + 1$. This function is total computable and should therefore be $\{f(m)\}$ for some m. $g(m) = \{f(m)\}(m) + 1 \neq \{f(m)\}(m)$, so $g \neq \{f(m)\}$. □

Theorem 6.2. Fix T, a recursively axiomatised system of arithmetic. $\{n\colon T \vdash \{\underline{n}\} \text{ total } \mathbb{N} \to \mathbb{N}\}$ is semi-decidable, but not decidable.

Proof. Semi-decidable by enumerating proofs.

Consider a machine $\mathcal{M}$ that tests, for each pair $\langle p, n\rangle$ of a T-proof p and a number n whether or not p is a T-proof that $\{\underline{n}\}$ is a total function $\mathbb{N} \to \mathbb{N}$. We obtain a volcano that emits all pairs $\langle p, n\rangle$ such that p is a T-proof that $\{n\}$ is total. Let $v(k)$ be the kth such pair emitted by the volcano. Let the T-bad function be

$$(\lambda k \in \mathbb{N})(\text{let } v(k) = \langle p, n\rangle \text{ in } \{\underline{n}\}(k) + 1)$$

This "diagonalises our volcano" and the set of functions proved by T to be total. Clearly T cannot prove that this function is total. □

Definition. An arithmetic theory T is **SOUND** if every theorem of T is true (in the standard model $\mathbb{N}$).

Theorem 6.3. Every sound recursively axiomatisable theory of arithmetic is incomplete.

Proof. If T is sound, then the T-bad function is total and T does not prove this. □

Definition. If there is an algorithm which given a semi-decidable subset $A \subseteq X$, the set of first-order arithmetic sentences true in a standard model emits a sentence in $X \backslash A$, then X is said to be **PRODUCTIVE**.

Theorem 6.4 (Gödel's (In)completeness Theorem). First-order logic is **COMPLETE**, however, it is not **DECIDABLE** if rich enough to express the workings of a Turing machine.

Proof. The set of sentences in first-order logic that are true in all structures is semi-decidable, being the set of theorems of a recursively axiomatizable theory.

Given machine $\mathcal{M}$ and i we can compute a formula φ which has the property that every model of φ is a complete course of computation of $\mathcal{M}(i)$ and has a last frame in which $\mathcal{M}$ has halted. If first-order logic is decidable, we can decide whether or not φ has a model. So we would be able to solve the Halting problem. □

Theorem 6.5 (Trakhtenbrot's Theorem). The set of sentences true in all finite structures is not complete.

Proof. Use the Halting problem. □

Chapter 7

Well-Quasi-Orders

Definition. A **QUASI-ORDER** (**QO**) $\langle X, \leqslant \rangle$ is transitive and reflexive. An infinite sequence $\langle x_i \rangle$ is

BAD. If there is no $i < j \in \mathbb{N}$ such that $x_i \leqslant x_j$.

GOOD. If it is not bad.

PERFECT. If $i \leqslant j$ implies $x_i \leqslant x_j$.

A **WELL-QUASI-ORDER** (**WQO**) is a QO with no bad sequences.

Proposition 7.1. $\langle X, \leqslant \rangle$ is WQO if and only if it has no ω-descending chains and no infinite antichains.

Lemma 7.2 (PERFECT SUBSEQUENCE LEMMA). Let $\langle X, \leqslant \rangle$ be WQO. Then every ω-sequence from X has a perfect subsequence.

Proof. $\Longrightarrow$ trivial. □

$\Longleftarrow$ Suppose $\langle x_i : i < \omega \rangle$ is a sequence from X. Two-colour the complete graph on $\mathbb{N}$, $\langle i, j \rangle$ is 0 if $i < j$, $x_i \leqslant x_j$ and 1 otherwise. By Ramsey there is a monochromatic infinite set. If of the second colour, we get an infinite anti-chain. If of the first colour it is a perfect subsequence, and thus $\langle x_i : i < \omega \rangle$ cannot be bad. □

Definition. Given a QO on X, it can be lifted to $\mathcal{P}(X)$ by $X' \leqslant X''$ if for all $x' \in X'$ there exists $x'' \in X''$ such that $x' \leqslant x''$.

It can also be lifted to $X^{<\omega}$ by setting $l_1 \leqslant l_2$ (**STRETCHES INTO**) if:

- l_1 is empty.
- $l_1 \leqslant \mathrm{tl}(t_2)$.
- $\mathrm{hd}(l_1) \leqslant \mathrm{hd}(l_2)$ and $\mathrm{tl}(l_1) \leqslant \mathrm{tl}(l_2)$.

Theorem 7.3. $\langle \mathcal{P}(X), \leqslant^+ \rangle$ is well-founded if and only if $\langle X, \leqslant \rangle$ is a WQO.

Proof. $\iff$ $\langle \mathcal{P}(X), \leqslant^+ \rangle$ is not well-founded.

$\iff$ There exists a descending sequence $X_0 >^+ X_1 >^+ X_2 >^+ \cdots$.

$\iff$ There is $(x_i)_{i \in \mathbb{N}}$ in X such that for all $i < j \in \mathbb{N}$ we have $x_i \not\leqslant x_j$.

$\iff$ X has a bad sequence.

$\iff$ X is not a WQO. □

Example. $\langle \mathbb{N}, = \rangle$ is well-founded, not a WQO. Lifts to $\langle \mathcal{P}(\mathbb{N}), \subseteq \rangle$ which is not well-founded: $\langle \{m \colon m > n\} \colon n < \omega \rangle$ is a descending sequence.

7.1 MBS Construction

Definition. A sequence $\langle x_i \rangle$ in X is a **MINIMAL BAD SEQUENCE** if x_n is a minimal member of

$$\{x \colon \exists \text{ bad sequence whose first } n \text{ members are } \langle x_0, \ldots, x_{n-1}, x \rangle\}.$$

Theorem 7.4. If X is well-founded but not a WQO, there is a minimal bad sequence.

Proof. Define $X_0 := \{x \in X \colon (\exists S)(S(0) = x \wedge S \text{ is bad})\}$ and

$$X_n := \{x \in X \colon (\exists S)(S(\overline{n}) = \overline{x} :: x \wedge S \text{ is bad})\}$$

inductively, setting x_n a minimal member of X_n using DC. □

Lemma 7.5 (Minimal Bad Sequence Lemma). Let:

- $\langle X, \leqslant \rangle$ be well-founded but not WQO.
- $B = \langle b_0, b_1, \ldots \rangle$ a MBS.
- $X' = \{x \in X \colon (\exists n)(x < b_n)\}$.

Then $\langle X', \leqslant \rangle$ is WQO.

Proof. Suppose that $S = \langle s_0, s_1, \ldots \rangle$ were a bad sequence from X'. We will prove by induction on $\mathbb{N}$ that nothing in S is $< b_n$.

Case 0. Consider s_i if $s_i < b_0$ then the tail of S, starting at s_i is bad, contradicting the choice of b_0 as minimal.

Case $n+1$**.** Suppose nothing in S smaller than any of $b_0, \ldots, b_n$, and suppose some $s_i < b_{n+1}$. Consider the sequence $b_0, \ldots, b_n$ and continues $s_i, s_{i+1}, \ldots$

It cannot be bad by minimality of b_{n+1}, and so it contains a good pair. Both S and B are bad, so the good pair must be $b_j \leqslant s_k$ with $j \leqslant n$ and $k \geqslant i$. Consider s_k, we must have m with $s_k < b_m$. But this m cannot be $\leqslant n$ by induction hypothesis. So $m > n$. But then $j \leqslant n < m$ with $b_j \leqslant b_m$. Contradicting badness of B. □

7.2 Kruskal's Theorem

Corollary 7.6. If $\langle X, \leqslant \rangle$ is well-founded QO, so is X-lists QO when ordered by stretching.

Proof. Suppose not, have an infinite descending sequence of X-lists under stretching. They can get shorter only finitely often, so we may assume that they all have the same length. The entries at each coefficient get smaller only finitely often, so must all be eventually constant. □

Lemma 7.7 (Higman's Lemma). If $\langle X, \leqslant \rangle$ is WQO, then $\langle X^{<\omega}, \text{stretching} \rangle$ is WQO.

Proof. Suppose not. We know it is well-founded, so we know that there is a minimal bad sequence $\langle a_i \rangle$ of lists. Look at the heads of the lists. These are WQO, so there must be a subsequence $\langle b_i \rangle$ of $\langle a_i \rangle$ by Lemma 7.2 such that the heads are perfect.

The bad sequence $\langle b_i \rangle$ now has heads who are increasing. Consider the tails. These are a WQO by Lemma 7.5, and so again we have that there is a subsequence $\langle c_i \rangle$ of $\langle b_i \rangle$ by Lemma 7.2 such that the tails are perfect.

In $\langle c_i \rangle$ the heads and tails are perfect, and so the sequence is perfect. But it is a subsequence of a bad sequence, so bad! Contradiction. □

Definition. An X-tree has the following recursive datatype:

- A single element of X is a **leaf** and a X-tree.
- A X-tree is otherwise made from an element of X and a list of X-trees.

If X is a QO, then if T, T' are X-trees, then $T \leqslant T'$ if:

- both are singleton trees $\{t\}$ and $\{t'\}$ and $t \leqslant t'$,
- $T \leqslant T''$ for some child T'' of T' or
- the root of T is less than or equal to the root of T' and the list of children of T is less than the children of T' by extending tree comparison to lists thereof.

Theorem 7.8 (Kruskal's Theorem). Finite trees over a WQO are WQO.

Proof. X-trees ordered by the extension of a QO on X is well-founded. In any descending sequence the skeletons are eventually constant (as they are finite objects). Consider the sequence limited to a certain point in the skeleton: as X is a well-order, these must eventually be constant. As there are finitely many, the sequence is eventually constant.

Suppose not WQO. Then there is a minimal bad sequence of trees $\langle a_i \rangle$. The roots are from a WQO, so there must be an increasing subsequence $\langle b_i \rangle$ of $\langle a_i \rangle$ such that the roots of b_i form a perfect sequence by Lemma 7.2. Let l_i be the list of children of b_i.

These trees in the lists form WQO by Lemma 7.5. Now use Lemma 7.7: lists over this WQO are WQO (under stretching). Now put the product order (point-wise) on $\langle \text{root}, \text{list of trees} \rangle$, and this product of WQOs is WQO, and so this sequence cannot be bad. Contradiction. □

Theorem 7.9 (Friedman's Finite Form). For all k, there exists an n such that if $T_1, \ldots, T_n$ is a list of trees where T_i has $k + i$ nodes, then there are $j < l$ such that $T_j \leqslant T_l$.

Proof. Consider the one-point WQO and suppose there is $k \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ there exists "bad finite sequence" of trees T_1^n, T_2^n, T_3^n, $\ldots$, T_n^n with $i < j < k$, $T_i^n \not\hookrightarrow T_j^n$ and T_i^n has $k + i$ nodes. Then we can find an infinite triangle of trees:

$$\begin{array}{ccccc} T_1^1 & & & & \\ T_1^2 & T_2^2 & & & \\ T_1^3 & T_2^3 & T_3^3 & & \\ \vdots & \vdots & \vdots & & \end{array}$$

Since in the first column, there are only finitely many trees with $k + 1$ nodes, so one tree, say T_1 appears infinitely many time. Now consider the rows that start with T_1, do the same with the second column. So $\langle T_i : i \in \mathbb{N} \rangle$ is a bad sequence, contradiction with Kruskal's Theorem. □

Chapter 8

Degree Theory

8.1 Many-One Reducibility

Definition. B is **MANY-ONE REDUCIBLE** to A, $B \leqslant_m A$ if there exists total computable function f such that for all $n \in \mathbb{N}$, $n \in B$ if and only if $f(n) \in A$.

$\mathbb{K} = \{\langle n, x\rangle : \{n\}(x)\downarrow\}$ is known as **THE HALTING SET** and so is $\{n : \{n\}(n)\downarrow\}$. Each of these is reducible to the other.

Lemma 8.1. $A \leqslant_m \mathbb{K}$ if and only if A is semi-decidable.

Proof. $\Longrightarrow$ Since $\mathbb{K}$ is semi-decidable it is the domain of a partial computable g. If $A \leqslant_m \mathbb{K}$ by virtue of some f then A is the domain of $g \circ f$ which makes A semi-decidable. □

$\Longleftarrow$ Let A be semi-decidable. Define a binary partial function f by $f(e, x) = \texttt{if } e \in A \texttt{ then } 1 \texttt{ else } \uparrow$. By Theorem 3.8 there is now a computable g such that $\forall x, e \in \mathbb{N}.\ \{g(e)\}(x) = f(e, x)$. From this we get $(\forall e)(\{g(e)\}(g(e))\downarrow \leftrightarrow e \in A)$ thus $(\forall e)(e \in A \leftrightarrow g(e) \in \mathbb{K})$ so $A \leqslant_m \mathbb{K}$ (diagonal Halting set) by virtue of g. □

8.2 Turing Reducibility

Definition. Present contents of register 2 to oracle, place the result from the oracle in register 3. We can enumerate the machines/programs of the new architectures, $\{e\}^A$ without specifying the oracle.

Relative computability using oracles $A \leqslant_T B$ if we can compute χ_A given χ_B as an oracle. These are quasi-orders, and by taking the intersection with its reversal we get the **Turing Equivalence**.

An equivalence class is a **Turing degree of undecidability**.

0 is the Turing degree of decidable sets. d' is the Turing degree of the halting problem for machines in d. Thus $0'$ is the degree of $\mathbb{K}$.

Theorem 8.2 (Kleene-Post Theorem). There are $\leqslant_T$-incomparable degrees $\leqslant_T 0'$.

Proof. Let us enumerate $\{0,1\}^{<\omega}$ as $\langle \eta_n : n \in \mathbb{N} \rangle$. Think of strings as functions from initial segments of $\mathbb{N}$ to $\{0,1\}$. We will find A, B such that $\{e\}^A$ and $\{e\}^B$ are $\leqslant_T$-incomparable. Observe:

- If $\{e\}^{\eta_\alpha}(x)\downarrow$ then $\{e\}^C(x)\downarrow$ for every $\eta_\alpha \subseteq C$.
- If $\{e\}^B(x)\downarrow$ then there is $a \in \mathbb{N}$, $\{e\}^{\eta_a}(x)\downarrow$.

We will diagonally construct A and B. We have two sequences of binary strings. $\langle \alpha_n : n \in \mathbb{N} \rangle$ such that α_{n+1} extends α_n and $\bigcup_{i\in\mathbb{N}} \alpha_i = \chi_A$. $\langle \beta_n : n \in \mathbb{N} \rangle$, β_{n+1} extends β_n and $\bigcup_{i\in\mathbb{N}} \beta_i = \chi_B$. Set $\alpha_0 = \beta_0 = \langle\rangle$.

Stage $2s+1$. Let x be the first number not in the domain of α_{2s}. If there are any η_α that are extensions of β_{2s} such that $\{s\}^{\eta_a}(x)\downarrow$ then use the least such a, and set α_{2s+1} to be $\alpha_{2s} :: y$ where y is the least element of $\{0,1\}\backslash\{\{s\}^{\eta_\alpha}(x)\}$. And set $\beta_{2s+1} = \eta_\alpha :: 0$. If there is no such η_a, then set $\alpha_{2s+1} = \alpha_{2s} :: 0$ and $\beta_{2s+1} = \beta_{2s} :: 0$.

Stage $2s+2$. Similarly, swap $\alpha \leftrightarrow \beta$ and $2s \leftrightarrow 2s+1$.

At stage $2s$, respectively stage $2s+1$: "$B \neq \{s\}^A$", "$A \neq \{s\}^B$". □

Theorem 8.3 (Friedberg-Munchnik Theorem). There are $\leqslant_T$-incomparable degrees of with representatives semi-decidable sets.

Chapter 9

Omitting Types

Definition. Fix language L, an n-**type** in L is a set $\Sigma = \{\sigma_i : i \in \mathbb{N}\}$ where each σ_i has n free variables.

Let $\mathcal{M}$ be a structure for L. We say $\mathcal{M}$ **realizes** Σ if and only if there is an n-tuple $\bar{c}$ such that $\mathcal{M} \models \sigma_i(\bar{c})$ for each i. Otherwise, Σ **omits** Σ.

A theory T **locally omits** a type Σ if whenever φ such that $T \vdash \varphi \to \sigma$ for all $\sigma \in \Sigma$ then $T \vdash \neg\varphi$.

For 0-types, we have by compactness that if for every finite subset $\Sigma' \subseteq \Sigma$, T has a model that realizes Σ'. Then T has a model that realizes Σ.

The standard model of arithmetic omits the 1-type $\{n \neq 0, n \neq S0, n \neq SS0, \dots\}$.

Theorem 9.1 (Propositional Omitting of Types). Let T be a propositional theory and $\Sigma \subseteq L(T)$ a type. If T locally omits Σ then there is a T-valuation omitting Σ.

Proof. By contrapositive. Suppose no valuation omits Σ. Then every formula in Σ is a theory of T, so take φ to be T, $T \vdash \varphi \to \sigma$ for every $\sigma \in \Sigma$ but $T \vdash \neg\varphi$. Contraposing, if $T \vdash \neg\varphi$ for every φ such that $(\forall \sigma \in \Sigma)(T \vdash \varphi \to \sigma)$ then there is a T-valuation omitting Σ. □

This can be extended to countably many types simultaneously, for propositional logic again.

Theorem 9.2 (Extending Omitting Types Theorem). Let T be a propositional theory, and for each $i \in \mathbb{N}$ let Σ_i be a type. If T locally omits each σ_i then there is a T-valuation omitting all the Σ_i.

Proof. We will show that, whenever $T \cup \{\neg A, \neg A_2, \ldots, \neg A_i\}$ is consistent, where $A_n \in \Sigma_n$ for $n \leqslant i$, then we can find $A_{i+1} \in \Sigma_{i+1}$ such that $T \cup \{\neg A_1, \ldots, \neg A_i, \neg A_{i+1}\}$ is consistent.

Suppose not. Then $T \vdash (\bigwedge_{1 \leqslant j \leqslant i} A_j) \to A_{i+1}$ for every $A_{i+1} \in \Sigma_{i+1}$. But, by assumption, T locally omits Σ_{i+1} so we must have $T \vdash \neg \bigwedge_{1 \leqslant j \leqslant i} A_j$ contradicting the consistency of A_i.

Now, as there is an enumeration of $L(T)$ we can start an structural process where, at each stage, we pick for A_{i+1} the first formula in Σ_{i+1} such that $T \cup \{\neg A_1, \ldots, \neg A_i, \neg A_{i+1}\}$ is consistent.

This gives a theory $T \cup \{\neg A_i : i \in \mathbb{N}\}$ which is consistent by compactness. Any valuation making this theory true omits every Σ_i. □

Chapter 10

Examples

10.1 Example Sheet 1

Exercise 10.1.1. Is there any significant difference between a natural number and its certificate? A countable ordinal and its certificate?

Solution. There need not be: if we have no extraneous information and the definition of certification and natural numbers are compatible.

A certificate of a countable ordinal can skip some at say limits.

Exercise 10.1.2 (Recursion Theorem). If $\langle X, R\rangle$ is a well-founded structure and $G\colon X \times V \to V$ then there is a unique f satisfying $(\forall x \in X)(f(x) = G(x, \{f(y)\colon R(y, x)\}))$.

Solution. Consider attempts: f with $\operatorname{dom} f \subset X$ and transitive and whenever $x \in \operatorname{dom} f$, then $f(x) = G(x, \{f(y)\colon R(y, x)\})$. If f, g are attempts, and $x \in \operatorname{dom} f \wedge \operatorname{dom} g$ then if $f(x) \neq g(x)$ there is least such, but then by definition of attempt $f(x) = g(x)$. So uniqueness evident.

If $\langle f_i\colon i \in I\rangle$ are attempts in a chain, so is $\bigcup_{i\in I} f_i$. So if there is x such that f is not in an attempt there is least such, but then we can take the union of g with $\operatorname{dom} g$ a subset of the transitive closure of $\{x\}$, and set $f(x) = G(x, \{f(y)\colon R(y, x)\})$. So we arrive at a contradiction.

So such f exists and is unique by taking the union of such attempts.

Exercise 10.1.3. (i) Consider the rectype of α-lists.

(a) What is primitive recursion on α-lists?

(b) Define stretching for α-lists by primitive recursion.

(ii) Consider the rectype of α-trees.

(a) Give a rectype declaration for the rectype of α-trees.

(b) What is primitive recursion on α-trees?

(c) Define stretching for α-trees by primitive recursion.

Solution. (i) (a) Initalize $f(\overline{x}, \emptyset) = z(\overline{x})$ and set

$$f(\overline{x}, \alpha :: l) = h(\overline{x}, \alpha, l, f(\overline{x}, l)).$$

(b) $f(l_2, l_1) = 1$ if l_2 stretches into l_1 and 0 otherwise by

$$\begin{aligned} f(l_2, \emptyset) &:= \texttt{if } l_2 = \emptyset \texttt{ then } 1 \texttt{ else } 0 \\ f(l_2, x :: l_1) &:= \texttt{if } l_2 = \emptyset \texttt{ then } 1 \texttt{ else} \\ &\qquad \texttt{if } x \leqslant y \texttt{ then } f(l_2', l_2) \text{ else } f(l_2', x :: l_1) \end{aligned}$$

where $l_2' = y :: l_2$.

(ii) (a) Every $x \in X$ is a tree. If $x \in X$ then x is a tree and l is a list of X-trees, then $x.l$ is a tree with root x and litter l.

(b) Initialize $f(\overline{x}, \alpha) = z(\overline{x}, \alpha)$ and set

$$f(\overline{x}, \alpha.l) = h(\overline{x}, \alpha, l, (f(\overline{x}, l_0), \ldots, f(\overline{x}, l_n))).$$

(c) $g(t_2, t_1) = 1$ if l_2 stretches into l_1 and 0 otherwise by

$$\begin{aligned} g(t_2, x_1) &:= \texttt{if } x < y = t_2 \texttt{ then } 1 \texttt{ else } 0 \\ g(t_2, y :: l_1) &:= \texttt{if } y = t_2 \texttt{ then } 0 \texttt{ else} \\ &\qquad \texttt{if } x \leqslant y \texttt{ then } f(l_2', l_2) \text{ else } f(l_2, [x.l_1]) \end{aligned}$$

where $t_2 = y.l_2$ and $[x.l_1]$ is the (X-tree)-list with one element $x.l_1$.

Exercise 10.1.4. Say that a machine **LOOPS** if and only if it repeats a configuration. Let LOOP $\subseteq \mathbb{N}$ be the set of all indices of machines that loop on input 0. What can you say about the decidability of LOOP?

Solution. Fix a computable bijection $f\colon \mathbb{N} \to \mathbb{N}^2$ and consider a machine $\mathcal{M}$ that and on input n:

(i) Set $c := 0$.

(ii) Compute $(t, i) := f(c)$.

(iii) Detect if a loop has occurred in $\{n\}_t(i)$ by tth step, if so, halt.

(iv) Set $c := c + 1$.

(v) Go to step (ii).

Step (iii) is possible due to the existence of a universal Turing machine and the possibility to compute data dumps. $\mathcal{M}$ halts if and only if $n \in$ LOOP and so LOOP is semi-decidable.

Suppose LOOP is decidable, fix $\mathcal{M}'$ such that $\mathcal{M}'$ in extension is χ_{LOOP}. Consider $\mathcal{M}''$ that on input n:

(i) Computes m such that $\{m\}$ is $\{n\}(n)$ in intension for all inputs.

(ii) $\mathcal{M}'(m) = \begin{cases} 1 & \text{halts} \\ 0 & \text{loops} \end{cases}$

Note in particular, $m \in$ LOOP if and only if $\{n\}(n)$ loops and so $\mathcal{M}''(n)$ halts if and only if $\{n\}(n)$ loops.

$\mathcal{M}'' = \{n\}$ for some $n \in \mathbb{N}$. Now $\{n\}(n)$ halts if and only if $\{n\}(n)$ loops, contradiction. So LOOP is not decidable.

Exercise 10.1.5. Turing machines are very robust under modification of their definition. Consider a TM with input, work and output that over alphabet $\{0, 1\}$ that can only write 1, but cannot write 0. Assume that output and work tape start with all 0s. Is this model as powerful as standard TMs?

Solution. Given a Turing machine in the original language. Any bit in the original is turned into 4 bits, set depending on whether:

(i) it is 0 or 1,

(ii) it is the start,

(iii) it is the end

(iv) it is copied already.

If a 0 is written in the original where there is currently a 1, or it tries to move to a place on the tape that is marked with "end" the following is a heuristic description of what the new Turing Machine does:

(i) remember current position and instruction somewhere to the right of the current end,

(ii) from the current position, go left until you reach the start,

(iii) iteratively copy each bit (except the 1 if we are in the “write 0” case) from the start until the end into a contiguous block to the right of the end with four empty bits in between,

(iv) in the new data, refind the original position and instruction, continue computation.

The original instructions are changed such that computation goes as normal, bar taking into account that moves need to be multiplied by 4.

Definition. We define a partial function $\delta\colon \mathrm{dom}(\delta) \to \mathrm{Cord}$ (where COrd is the set of countable ordinals) as follows:

(i) $\delta(0) = 0$,

(ii) $\delta(3^n) = \delta(n) + 1$,

(iii) $\delta(5^n) = \sup_{i\in\mathbb{N}} \delta(\{n\}(i))$, if $\{n\}$ is total and for all i, $\{n\}(i) \in \mathrm{dom}(\delta)$.

We call $\alpha \in \mathrm{COrd}$ is a **COMPUTABLE ORDINAL**, if and only if $\exists n\colon \delta(n) = \alpha$.

Exercise 10.1.6. Is there a partial computable function $p\colon \mathrm{dom}(\delta)^2 \to \mathbb{N}$ such that whenever n, $m \in \mathrm{dom}(\delta)$, then $\delta(p(n,m)) = \delta(n) + \delta(m)$?

Is there a recursively enumerable set A such that if $n \in \mathrm{dom}(\delta)$ then $n \in A$ if and only if $\delta(n)$ is a successor ordinal?

Solution. Define $p(n,m)$ by recursion as in the following pseudo-code:

(i) If $m = 0$ then return n.

(ii) If $m = 3^{m'}$ then return $3^{p(n,m')}$.

(iii) Set m' such that $m = 5^{m'}$.

(iv) Return m'' such that $\{m''\}(i) = p(n, 3^{\{m'\}(i)})$.

This last step is possible, since this $\{m''\}$ in a computer language allowing abstractions is computable without knowing the values of p. This shows it is Turing-computable by the Church-Turing Thesis.

The problem that remains is whether it halts for n, $m \in \mathrm{dom}(\delta)$. By recursion it does: let $H \subseteq \mathrm{dom}(\delta)$ be the m for which it halts. $0 \in H$. If $m \in H$ then $3^m \in H$. Suppose $m \in H$ and $5^m \in \mathrm{dom}(\delta)$, then $3^{\{m\}(i)} \in H$ for all i by definition and so $\{m'\}(i) = p(n, 3^{\{m\}(i)})$ is satisfiable by a computable function that can be computed.

The resolution of the last part is unknown.

10.2 Example Sheet 2

Exercise 10.2.1 (Past Tripos Question). An interleaving of two finite words w_1 and w_2 is obtained by inserting the letters of w_1 into w_2 in order (for example, both $a1b$ and $ab1$ are interleavings of ab and 1, but $b1a$ is not). Let $L_1 \oplus L_2$ denote the language containing all interleavings of words from L_1 with words from L_2.

(i) If L_1 and L_2 are regular, what about $L_1 \oplus L_2$?

(ii) If L_1 and L_2 are computable, what about $L_1 \oplus L_2$?

Solution. (i)

- A language L is regular if and only if it is recognized by a finite state machine. Suppose $\mathcal{M}_1$ and $\mathcal{M}_2$ are finite state machines recognising L_1 and L_2 respectively.
- By considering the natural finite state machine on the power set of the states of a non-deterministic state machine it follows that given a non-deterministic state machine there is finite state machine that recognizes the same language.

Consider the non-deterministic finite state machine $\mathcal{M}$ with: states the Cartesian product of the states of $\mathcal{M}_1$ and $\mathcal{M}_2$ and:

Transition Table. Transition $(s_1, s_2) \xrightarrow{t} (s_1', s_2')$ allowed if

- $s_1 = s_1'$ and s_2 transitions to s_2' by character t.
- $s_2 = s_2'$ and s_1 transitions to s_1' by character t.

Accepted States. (s_1, s_2) accepted if s_1 accepted in $\mathcal{M}_1$ and s_2 accepted in $\mathcal{M}_2$.

That is, $\mathcal{M}$ is the machine that at any steps either progresses by taking a step in the states of $\mathcal{M}_1$ or $\mathcal{M}_2$. By our imported results, it only remains to show that $\mathcal{M}$ recognizes $L_1 \oplus L_2$.

Suppose w in our language and fix a run of $\mathcal{M}$ on this input. Let w_1 be the string obtained by stringing together the characters t that have changed the state of the first coordinate of $\mathcal{M}$ in this run. Similarly for w_2 and the second coordinate. By construction, w is accepted by this run if and only if w_1 is accepted by $\mathcal{M}_1$ and w_1 is accepted by $\mathcal{M}_2$.

The w_1 and w_2 that occur are exactly those that interleave to give w, and thus w is recognized by $\mathcal{M}$ if and only if w is the interleaving of w_1 and w_2 which are accepted by $\mathcal{M}_1$, $\mathcal{M}_2$ respectively.

(ii) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ compute L_1 and L_2 respectively. Note that for any word w, there are only finitely many words w_1 and w_2 it could be an interleaving of, and these can be effectively computed, denoted by w_{1_n} and w_{2_n} and let N_1 and N_2 be how many there are. Let $\mathcal{M}$ be a Turing machine working as follows on input w:

(a) Set $n_1 := 0$.
(b) Set $n_2 := 0$.
(c) If $\mathcal{M}_1(w_{1_{n_1}}) = \mathcal{M}_2(w_{2_{n_2}}) = 1$, output 1.
(d) Set $n_2 := n_2 + 1$.
(e) If $n_2 \leqslant N_2$ go to (b).
(f) Set $n_1 := n_1 + 1$.
(g) If $n_1 \leqslant N_1$ go to (a).
(h) Output 0.

Then $\mathcal{M}$ shows $L_1 \oplus L_2$ computable.

Exercise 10.2.2. Will the same argument as for Trakhtenbrot's Theorem show that:

(i) The sentences true in arbitrarily large finite models,
(ii) The sentences true in all sufficiently large models,
(iii) The sentences true in all infinite models,
(iv) The sentences true in all finite models that have even cardinality and all infinite models.

is not semi-decidable?

Solution. (i) Unkown.

(ii) A sentence φ is true in all sufficiently large models if and only if there exists an n such that $\varphi_n \vdash \varphi$. The consequences of φ_n can be enumerated, and so by using a bijection $\mathbb{N}^2 \to \mathbb{N}$ we can enumerate the union of these consequences.

(iii) If φ is true in all infinite models then $\neg\varphi$ has no infinite models; so $\neg\varphi$ does not have arbitrarily large finite models by Compactness. So $\neg\varphi$ can be refuted from the set of axioms

$$\varphi_n := \exists x_1 \ldots \exists x_n \colon \bigwedge_{i<j} x_i \neq j.$$

So φ follows from this scheme, and thus we can enumerate the formulas with arbitrarily large finite models by enumerating the consequences of this scheme. This is possible as the scheme itself is enumerable.

(iv) Set

$$\psi_n := \exists x_1 \dots \exists x_n \colon \bigwedge_{i<j} x_i \neq j \wedge \forall x \colon \bigvee_i x = x_i.$$

$\mathcal{M} \models \psi_n$ if and only if $|M| = n$. Can enumerate all consequences of ψ_{2n} separately for $n \in \mathbb{N}$ and for infinite models by (iii). Use another bijection $\mathbb{N} \to \mathbb{N}^2$ to find when they have found the same consequence, and then output this. This outputs all such sentences at some stage.

Exercise 10.2.3. What goes wrong if *finitely* is removed (replaced by infinitely) in the following?

(i) The pointwise product of finitely many WQOs is WQO.

(ii) The intersection of finitely many WQOs is WQO.

(iii) The disjoint union of finitely many WQOs is WQO.

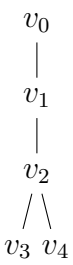
Solution. (i) Consider the ω product of the WQO on $\{0,1\}$ given by the usual order. $\{0,1\}^\omega$ is not a WQO, as $10\dots$, $010\dots$, $0010\dots$ form an infinite antichain.

(ii) Consider the WQOs $<_n$ on $\{0,1\}^\omega$ given by $\langle x_i \rangle <_n \langle y_i \rangle$ if $x_i < y_i$ for all $i < n$. $<_n$ has no bad sequences as $\{0,1\}^n$ is WQO. Consider the intersection, which is the same as the induced order on $\{0,1\}^\omega$ which is not a WQO.

(iii) This goes wrong in *any* infinite case. Given $\langle X_i \colon i \in I \rangle$ with $x_i \in X_i$ WQO for all $i \in I$ and I infinite. Consider $X = \coprod_{i \in I} X_i$. Without loss of generality $X_i \subseteq X$, then $\{x_i \colon i \in I\}$ is an antichain. As I is infinite, X cannot be a WQO.

Exercise 10.2.4 (Past tripos question). Suppose we quasi-order finite trees as follows: $T \preccurlyeq T'$ if there is an injection from the vertex set of T to the vertex set of T' that preserves the root and preserves adjacency. Is this a WQO?

Solution. Let T_n be a tree with $n+3$ vertices. The vertices are $v_0, \ldots, v_{n+2}$ with v_0 being the root. The edges are exactly: $v_i v_{i+1}$ for $0 \leqslant i \leqslant n$ and $v_n v_{n+2}$. So for example T_2 is:



Consider an injection of vertices v_k from T_i to vertices w_k from T_j for $i < j$ that preserves the root and adjacency. By preserving root we have that $v_0 \mapsto w_0$. By preserving adjacency we find by induction that $v_k \mapsto w_k$ for $0 \leqslant k \leqslant i+1$. By injectivity, v_{k+2} needs to be mapped to w_k for $i+1 < k$, however none of these are adjacent to w_i and thus adjacency cannot be preserved. Contradiction.

Thus T_n is a bad sequence, and thus this order cannot be a WQO.

Exercise 10.2.5. We say that a function $f\colon \mathbb{N} \to \mathbb{N}$ *grows too fast to be computable*, if for any function $g\colon \mathbb{N} \to \mathbb{N}$ we find that if $f(n) \leqslant g(n)$ eventually then g is not computable. Give an explicit construction of a function that grows too fast to be computable, or prove that there is no such function.

Solution. Enumerate the total computable functions $f_1, \ldots$ then set

$$f(n) = \max_{1 \leqslant i \leqslant n} f_i(n) + 1.$$

Then $f_i(n) < f(n)$ eventually for all i. Thus $f(n) \leqslant g(n)$ eventually implies $g(n)$ not computable.