

Table of Contents

1	The Anatomy of a Turing Machine	1–1
1.1	Formality	1–4
1.2	The Turing machine as map	1–4
1.3	The Church-Turing Thesis	1–5
2	Prefix-free codes	2–1
2.1	Domain of definition	2–1
2.2	Prefix-free sets	2–2
2.3	Prefix-free codes	2–3
2.4	Standard encodings	2–3
2.4.1	Strings	2–3
2.4.2	Natural numbers	2–4
2.4.3	Turing machines	2–5
2.4.4	Product sets	2–6
2.4.5	A second code for $\mathbb{N}$	2–6
3	Universal Machines	3–1
3.1	Universal Turing machines	3–1
4	Algorithmic Entropy	4–1
4.1	The entropy of a string	4–1
4.2	Entropy of a number	4–3
4.3	Equivalent codes	4–4
4.3.1	The binary code for numbers	4–5
4.4	Joint entropy	4–5
4.5	Conditional entropy	4–6
5	The Halting Problem	5–1
5.1	Sometimes it <i>is</i> possible	5–1
5.2	The Halting Theorem	5–2

6	Recursive sets	6-1
6.1	Recursive sets	6-1
6.1.1	Recursive codes	6-2
6.2	Recursively enumerable sets	6-2
6.3	The main theorem	6-4
7	Kraft's Inequality and its Converse	7-1
7.1	Kraft's inequality	7-1
7.1.1	Consequences of Kraft's inequality	7-3
7.2	The converse of Kraft's inequality	7-4
7.3	Chaitin's lemma	7-7
8	A Statistical Interpretation of Algorithmic Entropy	8-1
8.1	Statistical Algorithmic Entropy	8-1
8.2	The Turing machine as random generator	8-3
9	Equivalence of the Two Entropies	9-1
10	Conditional entropy re-visited	10-1
10.1	Conditional Entropy	10-1
10.2	The last piece of the jigsaw	10-2
10.3	The easy part	10-2
10.4	The hard part	10-3
10	Stack machines	10-9
10.1	Stacks	10-9
10.2	The anatomy of a stack machine	10-9
10.3	A toy language	10-10
10.3.1	An example	10-11
10.3.2	Labels	10-11
10.4	From program to machine	10-12
10.5	From machine to program	10-13
10.6	Two stacks suffice	10-14
10.6.1	Divide and conquer	10-15
10.7	Two stacks suffice	10-16
11	Universal Machines	11-1
11.1	Universal Turing machines	11-1
11.2	Construction of a universal machine	11-1
11.2.1	Equivalence of Turing and 2-stack machines	11-1
11.3	Storing the rules	11-3
11.4	Reading in the rules	11-4

11.5	An overview of our universal machine	11-4
11.5.1	Finding the appropriate rule	11-4
A	Cardinality	1-1
A.1	Cardinality	1-1
A.1.1	Cardinal arithmetic	1-2
A.2	The Schröder-Bernstein Theorem	1-2
A.3	Cantor's Theorem	1-4
A.4	Comparability	1-5
A.4.1	The Well Ordering Theorem	1-10
B	An exercise in Turing programming	2-1
B.1	Introduction	2-1
B.2	The reading phase	2-3
B.3	The conversion phase	2-4
B.4	The writing phase	2-7
B	Programming a Turing machine	2-8
B.0.1	An example	2-9
B.1	From program to machine	2-9
B.2	From machine to program	2-10

Chapter 3

Universal Machines

CERTAIN TURING MACHINES have the remarkable property that they can emulate, or imitate, all others.

3.1 Universal Turing machines

We have defined a code $\langle T \rangle$ for Turing machines in the last Chapter (Subsection 2.4.3).

Definition 3.1. *We say that the Turing machine U is universal if*

$$U(\langle T \rangle p) = T(p)$$

for every Turing machine T and every string $p \in \mathbb{S}$.

Informally, a universal Turing machine can ‘emulate’ any Turing machine.

There is another definition of a universal machine which is sometimes seen: the machine U is said to be universal if given any Turing machine T we can find a string $s = s(T)$ such that

$$U(sp) = T(p)$$

for all $p \in \mathbb{S}$. Evidently a universal machine according to our definition is universal in this sense; and since we are only concerned with the *existence* of a universal machine it does not matter for our purposes which definition is taken.

Theorem 3.1. *There exists a universal Turing machine.*

We shall outline the construction of a universal machine in Chapter ??
The construction—which is rather complicated—can be divided into 4 parts:

1. We introduce a variant of Turing machines, using stacks instead of a tape.
2. We show that ‘2 stacks suffice’, with these stacks replacing the left and right halves of the tape.
3. We show that $n \geq 2$ stacks is equivalent to 2 stacks, in the sense that given an n -stack machine S_n we can always find a 2-stack machine S_2 such that

$$S_2(p) = S_n(p)$$

for all inputs p .

4. We show that a universal machine can be implemented as a 4-stack machine. 2 of the stacks being used to store the rules of the machine being emulated.

But for the moment we merely point out that the Church-Turing thesis suggests that such a machine must exist; for it is evident that given the rules defining a machine T , and an input string p , the ‘human computer’ can determine the state of the machine and the content of the tape at each moment $t = 0, 1, 2, \dots$, applying the appropriate rule to determine the state and tape-content at the subsequent moment.

There are many universal machines. Indeed, a universal machine U can emulate itself. So the machine V defined by

$$V(p) = U(\langle U \rangle p)$$

is also universal.

It is an interesting question to ask if there is a *best* universal machine in some sense of ‘best’. One might ask for the universal machine with the minimal number of states; for there are only a finite number of Turing machines with $\leq n$ states, since there are only a finite number of rules

$$(q_i, b) \mapsto (a, q_o)$$

with $0 \leq q_i, q_o \leq n$. This question has some relevance for us, since we shall define algorithmic entropy (shortly) *with respect to a given universal machine*. It will follow from the fact that two universal machines U, V can emulate each other that the entropies with respect to U and V cannot differ by more than a constant $C = C(U, V)$. But it would be nice to have a concept of *absolute* algorithmic entropy.

Summary

The universal machine U performs exactly like the machine T , provided the program is prefixed by the code $\langle T \rangle$ for the machine:

$$U(\langle T \rangle p) = T(p).$$