ON COMPUTAMBLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHENIDUGSPROBLEM

Turing 1936

Where are We?

Ignoramus et ignorabimus

We do not know
We shall not know

Wir mussen wissen
Wir werden wissen

We must know
We will know

The right side is where we wish to be, but sometimes we are on the left side.

Entscheidungsproblem (“Decision Problem”)

The Entscheidungsproblem asks for
an algorithm that takes as input a statement of a first-order-logic (possibly with a finite number of axioms beyond the usual axioms of first-order logic) and answers "Yes" or "No" according to whether the statement is *universally valid.*

Turing will later show this problem is undecidable.

Some Definitions

- **Σ** - is a finite set of symbols
- **“First Kind Symbols”** – 1/0
- **“Second Kind Symbols”**- special symbols (#,*,@,% etc.)
- **M(x)** – a machine a.k.a “turing-machine”. (x) is the machine input
- **a-machine** - “automatic-machine” is a deterministic turing-machine. For the scope of this lecture any ‘machine’ is an a-machine.
- **c-machine** - “choice-machine” is a non-deterministic turing-machine

More Definitions

- **“Circle-free machine”**- prints an *infinite* number of symbols of the first kind
- **“Circular machine”**- prints a *finite* number of symbols of the first kind
- We shall assume a machine prints only a **finite** number of symbols of the **second kind** therefore a machine **halts** iff its **circular**.
- “**Computable sequence**” – the sequence of 1/0s written by the machine. Can be either finite or infinite. It represents a “**Computable number**”.

S.D and D.N

- **S.D** is the “standard description” of a machine. Basically it is a shortened version of the table that represents the machine’s *transition function.*
- **D.N** is the “description number” of a machine. It is a natural number that can be obtained directly from the machine S.D.
- The mapping between a machine to it’s S.D to it’s D.N is both one to one and onto. Therefore the set of all machines is enumerable.
- <M> - a notation for the D.N of M where M is a machine.

The Universal Machine

- We use the notation U(<M>,x) to represent the "Universal Machine".
- Basically its an a-machine that, given <M> and an input x, can simulate the run of M on x.
- U obtains a full description M from its D.N and performs the required actions to run M on x as described.
- One can think of U as a 'computer' that can run any 'computer program' M on any input x.

The Set of Computable Sequences is Enumerable

For each computable sequence α we can match S_α a non-empty set of the D.Ns of the machines that compute α.

We know that S_α is not empty because α is a computable sequence and therefore there is at least one machine that computes α.

Moreover for each $\alpha_1 \neq \alpha_1$: $S_{\alpha_1} \cap S_{\alpha_2} = \emptyset$. This is because a deterministic machine cannot compute more then one sequence.

We already showed that the set of all machines is enumerable. Therefore the set of all computable functions is enumerable.

HALT is Undecidable
“Proof in Two Minutes”

- Using common, university course, notations.
- Assume for the sake of obtaining a contradiction that there is a machine H(<M>,x) that outputs 1 iff M halts on x.
- We define a machine D(x) that given any input x runs H(<D>,x). If H(<D>,x)=1, D goes into an infinite loop. Otherwise D halts.
- What will D(<D>) do? Will it halt?

HALT is Undecidable (cont. I)

- We will prove that determining whether a machine is circular or circle-free is undecidable.
- Previously we showed that the set of computable sequences is enumerable. It can be easily shown that set of infinite computable sequences (i.e. the set of sequences computed by circle free machines) is also enumerable. Let $\{\varphi_n\}_n$ be such enumeration whereas φ_n is the n-th sequence.

HALT is Undecidable (cont. II)

- We show that the diagonal of this enumeration β is not computable.
- Otherwise for every $n \in N$ we could have obtained $\beta'(n) = 1 - \beta(n)$. Clearly β' is computable from β and therefore there must be $k \in N$ s.t $\varphi_k = \beta'$
- But then of course $\varphi_k(k) \neq \beta'(k)$ thus obtaining a contradiction to the enumerability of the set of computable sequences. Therefore β isn't computable sequence.

HALT is Undecidable (cont. III)

- Assuming the existence of a machine ‘H’ that can decide given the D.N of another machine ‘M’ whether ‘M’ is circle-free or circular we will show a machine ‘D’ that computes β thus obtaining a contradiction.
- D will run in sections 1…N… each time computing R(N) which is the number of D.Ns describing a circle-free machine in the range 1…N. In section N D will consult with H whether N is the D.N of a circular or a circle-free machine.

HALT is Undecidable (cont. IV)

- If M(N) is circular R(N)=R(N-1) and D will continue to the N+1 section.
- Otherwise R(N)=R(N-1)+1 and D computes the R(N) digit of the sequence computed by M(N) and writes it on the tape.
- Now it is clear that D computes β on it's tape. Therefore 'D' cannot exist and so also 'H'.
- We showed that there can be no machine that decides whether some other machine is circular or circle-free.

Why talk about 'Computability'?

- The definition captures the essence of computation. This is arguable yet convincing.
- Many things are computable according to our definition. For example π,e are computable, and so are the roots of any polynom with rational coefficients - "algebraic numbers".
- It was shown that other computational models (for example λ-calculus and definable sequences) are equivalent in power to a turing-machine .

‘Definable’ is also ‘Computable’

- We introduce another definition for computability using concepts of mathematical logic.
- Let α be a sequence of 0/1s. Let σ be a signature . σ includes ‘F’- the successor function, ‘u’- a constant symbol and $G_\alpha(t)$ - a predicate that evaluates to TRUE iff the ν$[t]$ digit of α is 1 (assuming our domain is N). We will say that α is ‘definable’ if there exists some provable formula ‘U’ over σ s.t for every $n \in N$ exactly one of the following two formulae is provable:

$$A_n \coloneqq U \& F^{(n)} \rightarrow G_\alpha(u^{(n)})$$

$$B_n \coloneqq U \& F^{(n)} \rightarrow -G_\alpha(u^{(n)})$$

‘Definable’ is also ‘Computable’ (cont. I)

- Now we want to show that any ‘definable’ sequence α is also ‘computable’.
- A sketch of the proof. Let α be a definable sequence. Let ‘n’ be an arbitrary positive integer. We wish to compute α (n), the n-th digit of α.
- In order to do so we need to prove either A_n or B_n. We know that exactly one of them is provable (by definition of computable sequence). We set about to find its proof.
- Finding the proof is straight forward. We enumerate over all the proofs of W.F.F over σ .

‘Definable’ is also ‘Computable’ (cont. II)

- First we obtain proofs of length 1. These are simply the axioms. Then we obtain proofs of length 2 by trying to apply all the derivation rules on each of the axioms. And so on.
- We already know that either A_n or B_n is provable by a proof sequence of some length K. Sooner or later we will find all the proofs of length K. One of them will be the proof of either A_n or B_n.
- It can be easily shown (from the definition of G_α) that α (n)==1 iff A_n is provable.

Back to Entscheidungsproblem

- In the last section of the article Turing shows that the "Decision Problem" is undecidable.
- It is shown that $\mathrm{E.P} \geq HALT$. This is called 'reduction' and it is used extensively throughout the article.
- "If E.P is decidable then so is HALT. We know that HALT is undecidable and therefore E.P is also undecidable. "

Relation to Incompleteness Theorem

- Godel Incompleteness Theorem states that: "*Any effectively generated theory capable of expressing elementary arithmetic cannot be both consistent and complete.*"
- It was later shown by Stephen Kleene that the existence of a complete effective theory of arithmetic with certain consistency properties would force the halting problem to be decidable, a contradiction.