

HISTORY OF ALGORITHMS

- Euclid algorithm for GCD
- Algorithms for computing functions and constants
- Simplex algorithms

Hilbert's 10th problem (1900):

Find an algorithm to decide, given an integer polynomial, whether it has any integer roots.

Hilbert's Entscheidungsproblem (1928):

Find an algorithm to decide whether a given sentence in first-order logic is logically valid.

A positive solution to Entscheidungsproblem would give a positive solution to the 10th problem.

Turing (1936) & Church (1936):

No problem to solve Entscheidungsproblem exists.

The Entscheidungsproblem is undecidable.

An essential ingredient to the above was to give a precise definition of what it means for a problem to be computable / have an algorithm.

Turing: Turing machine

(On computable numbers and Entscheidungsproblem)

Church: λ -calculus

(An unsolvable problem of elementary number theory)

DEFINITION OF AN ALGORITHM

Algorithm should be a **finite** description of a computation process.

It should specify the computation process fully.

The process of following an algorithm should not need any creative input.

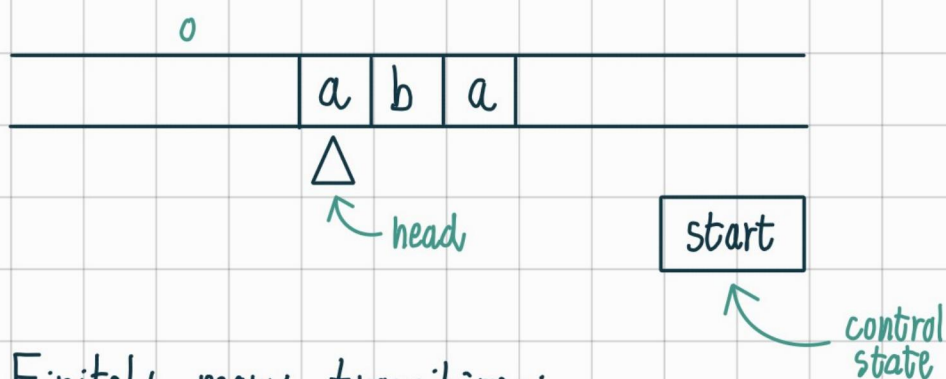
So the process of following an algorithm can be carried out by a machine.

The machine performs steps in time using working space to perform calculations.

For maximum generality, we do not bound time and space in advance.

TURING MACHINES

Example:



Finitely many transitions:

$$(\text{start}, a) \rightarrow (\text{start}, b, +1)$$

$$(\text{start}, b) \rightarrow (\text{start}, a, +1)$$

$$(\text{start}, \square) \rightarrow (\text{return}, \square, -1)$$

$$(\text{return}, a) \rightarrow (\text{return}, a, -1)$$

$$(\text{return}, b) \rightarrow (\text{return}, b, -1)$$

$$(\text{return}, \square) \rightarrow (\text{halt}, \square, +1)$$

$$\Gamma = \{\square, a, b\}$$

$$Q = \{\text{start}, \text{return}, \text{halt}\}$$

A Turing machine (TM) is given by (Γ, Q, γ) , where:

- Γ is a finite set (the tape alphabet) with $\square \in \Gamma$
- Q is a finite set (the (control) states) with $\text{start} \in Q$
- γ is a partial function $Q \times \Gamma \rightarrow Q \times \Gamma \times \{-1, 0, 1\}$ (the transition function)

A control state $q \in Q$ is halting if there is no element $(q, a) \in \text{dom}(\gamma)$ (there is no transition defined on state q).

We run a TM on an initial tape configuration $\mathbb{Z} \rightarrow \Gamma$ such that only finitely many are not blank, from an initial head position $i \in \mathbb{Z}$, and starting from the start state.

So, we run the machine from an initial machine configuration:

(t, i, start)
 $\uparrow$ $\uparrow$ $\uparrow$
 tape config head position control state

Computation process is following our transitions:

$$(t_0, i_0, \text{start}) \rightarrow (t_1, i_1, q_1) \rightarrow \dots \rightarrow (t_n, i_n, q_n)$$

TM_s COMPUTING FUNCTIONS

$$\Sigma_1^* \rightarrow \Sigma_2^*$$

Σ_1, Σ_2 alphabets (finite sets)

Σ^* the set of words over Σ (finite sequences)

ϵ empty word

A TM is a $\epsilon_1 \rightarrow \epsilon_2$ word transducer if:

- $\Sigma_1 \cup \Sigma_2 \subseteq \Gamma \setminus \{\square\}$
- $\text{halt} \in Q$ where halt is a halting state

It computes the partial function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ if:

- For any $w \in \Sigma_1^*$ with $w \in \text{dom}(f)$, if the machine is in the form $\frac{\text{///} \mid w \mid \text{///}}{\Delta}$ in the start state, the execution terminates in $\frac{\text{///} \mid f(w) \mid \text{///}}{\Delta}$ in the halt control state.
- For any $w \in \Sigma_1^*$ with $w \notin \text{dom}(f)$, $-||-$, the computation does not terminate in the halt control state.

Function $f: \Sigma_1^* \rightarrow \Sigma_2^*$ is *computable*, if there is some TM that computes it.

Example: Our TM from above flips each symbol.

A *k-tape TM* is (Γ, Q, γ) , where $\gamma: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{-1, 0, 1\}^k$.

TM's RECOGNISING LANGUAGES

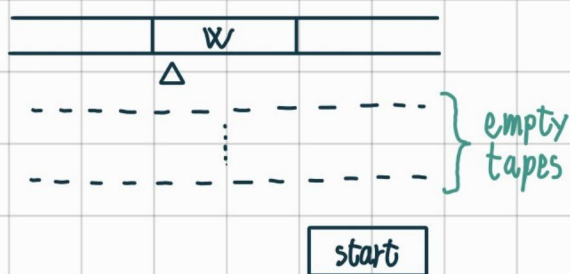
A *language* over Σ is a subset $L \subseteq \Sigma^*$.

A TM is *suitable* for language recognition over Σ if:

- $\Sigma \subseteq \Gamma \setminus \{\square\}$
- Q contains two distinguished halting states accept, reject

A TM *recognises/semidecides* $L \subseteq \Sigma^*$ if:

- For any $w \in L$ if the TM is run on configuration



it terminates in the accept state.

- For any $w \in \Sigma^* \setminus L$, $-||-$, the machine does not terminate in the accept state.

Language L is **semi decidable** if it is semidecided by some TM.

Similarly, a TM **decides** L if for $w \in \Sigma \setminus L$ it instead terminates in the reject state.

Language L is **decidable** if it is decided by some TM.

L decidable $\Rightarrow L$ semidecidable

There exist languages that are not semidecidable over $\Sigma \neq \emptyset$:

$|\Sigma^*| = \aleph_0$ (there are countably infinitely many words)

$|P(\Sigma^*)| = 2^{\aleph_0}$ (as the set of all languages over Σ is $P(\Sigma^*)$)

There are countably many TMs up to equivalence.

Hence only countably many semidecidable languages over Σ .

So there are $2^{\aleph_0}$ languages that are not semidecidable.

ENCODING TMs AS WORDS

We will encode 1-tape TMs as words.

We could use $\Sigma = \{0, 1\}$. Instead for convenience we use $\Sigma_u = \{0, 1, -, [,], \parallel, \cdot\}$.

We will encode a TM M as a word $\langle M \rangle \in \Sigma_u^*$.

$M = (\Gamma, Q, \delta)$

We assume $\text{start}, \text{accept}, \text{reject} \in Q$.

We encode a state q as word $\langle q \rangle \in \{-1, 0, 1\}^l$ for some $l \geq \log_3 |Q|$.

We assume $\begin{aligned} \langle \text{start} \rangle &= \overbrace{0 \dots 0}^l \\ \langle \text{accept} \rangle &= 1 \dots 1 \\ \langle \text{reject} \rangle &= -1 \dots -1 \end{aligned}$

For $a \in \Gamma$ we encode $\langle a \rangle \in \{-1, 0, 1\}^m$ where $m \geq \log_3 |\Gamma|$.

We assume $\langle \square \rangle = \overbrace{0 \dots 0}^m$.

For every instruction $\delta(q, a) = (q', b, d)$, $q, q' \in Q$, $a, b \in \Gamma$, $d \in \{-1, 0, 1\}$, the individual instruction is encoded as the word

$$[\langle q \rangle \cdot \langle a \rangle \parallel \langle q' \rangle \cdot \langle b \rangle \cdot d] \in \Sigma_n^*$$

of length $2l + 2m + 7$.

We then encode $M = (\Gamma, Q, \delta)$ as

$$\langle M \rangle := \langle \text{start} \rangle \cdot \langle \square \rangle \underbrace{[\langle q_1 \rangle \cdot \langle a_1 \rangle \parallel \langle q_1' \rangle \cdot \langle b_1 \rangle \cdot d_1] \dots}_{\text{encoding of instructions}}$$

And we encode word $w = w_0 \dots w_{n-1} \in \Sigma^*$ as

$$\langle w \rangle := \langle w_0 \rangle \cdot \dots \cdot \langle w_{n-1} \rangle$$

There exist languages that are undecidable but semidecidable.

$$L_{\text{accept}} \subseteq \Sigma_u^*$$

$$L_{\text{accept}} := \{ \langle M \rangle \cdot \langle w \rangle ; M \text{ is TM with tape alphabet } \Gamma, w \in (\Gamma \setminus \{\square\})^*, M \text{ accepts input } w \}$$

Theorem: L_{accept} is undecidable.

Proof: Suppose D is TM with input alphabet Σ_u that decides L_{accept} .

Modify D to construct a new TM N over input alphabet Σ_u that executes as follows:

- N reads its input word v and converts this to the word $v \cdot \langle v \rangle$ and returns the head to the start of $v \cdot \langle v \rangle$.
- Now execute D with input $v \cdot \langle v \rangle$, but swap the accept and reject states.

So:

- The execution of N will accept it and only if D rejects $v \cdot \langle v \rangle$.
- The execution of N will reject it and only if D accepts $v \cdot \langle v \rangle$.

What does N do if we give it $\langle M \rangle$ as input?

- N accepts $\langle M \rangle$ iff D rejects $\langle M \rangle \cdot \langle \langle M \rangle \rangle$
iff $\langle M \rangle \cdot \langle \langle M \rangle \rangle \notin L_{\text{accept}}$ iff M does not accept $\langle M \rangle$

- N rejects $\langle M \rangle$ iff D accepts $\langle M \rangle \cdot \langle \langle M \rangle \rangle$
iff $\langle M \rangle \cdot \langle \langle M \rangle \rangle \notin L_{\text{accept}}$ iff M accepts $\langle M \rangle$

We give N the word $\langle N \rangle$ as input.

N accepts $\langle N \rangle$ iff N does not accept $\langle N \rangle$



How might we semidecide L_{accept} ?

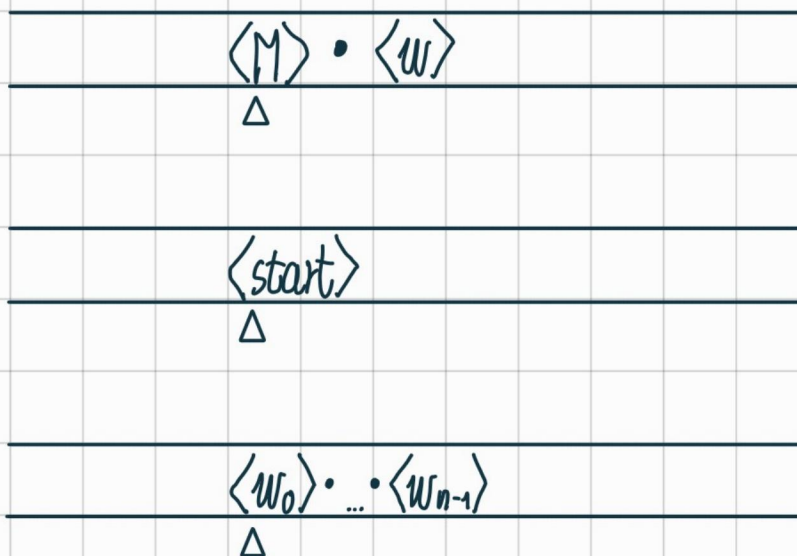
We shall do this using a universal TM.

For us a universal TM U will behave as follows:

- It takes as input strings of the form $\langle M \rangle \cdot \langle w \rangle$.
- It simulates the execution of M on input w .
- U ends in accept/reject state iff the simulation of M on w does the same

Theorem: There exists such a universal TM.

We can have a 3-tape universal TM:



Theorem: L_{accept} is semidecidable.

Proof: Our semideciding machine takes as input $v \in \Sigma_u^*$, then:

- Parses v to see if it has the form $\langle M \rangle \cdot \langle w \rangle$ for some TM M and $w \in (\Gamma_m \setminus \{\square\})^*$.
- Returns the head to beginning of tape.
- Behaves like the UTM.

Other similar semidecidable but not decidable languages:

$$L_{\text{halt}} := \{ \langle M \rangle \cdot \langle w \rangle \mid M \text{ halts on input } w \}$$

(The halting problem is undecidable, but semidecidable)

$$L_{\text{may halt}} := \{ \langle M \rangle \mid \exists w \in (\Gamma_m \setminus \{\square\})^* : M \text{ halts on } w \}$$

Our UTM U simulates TMs configured for language recognition.

One can also define a UTM for TMs configured for function computation:

This U takes as input $\langle M \rangle \cdot \langle w \rangle$, where we encode $\langle \text{halt} \rangle$ in some specified way.

Theorem: On input $\langle M \rangle \cdot \langle w \rangle$, this U will:

- Halt if and only if M halts on input w .

- Halt in the halt state if and only if M halts in the halt state on input w .
- Halt in the halt state with the result of M 's computation as the word on tape 1. We can leave the word in the encoded form.

