

Universal Turing Machines

Takes as input a description of another TM M and an input x for M and simulates the behavior of M on x , i.e.,

$$U(M; x) = M(x).$$

Universal Turing Machine

We encode the symbols and states of TM M by integers (in binary).

Let $M = (K, \Sigma, \delta, s)$. Then we assume

$\Sigma = \{1, 2, \dots, |\Sigma|\}$ and $K = \{|\Sigma| + 1, |\Sigma| + 2, \dots, |\Sigma| + |K|\}$

s is $|\Sigma| + 1$.

Special things like $\rightarrow, \leftarrow, -, yes, no$ are encoded by integers from $|K| + |\Sigma| + 6$.

All numbers will be in binary with $\lceil \log(|K| + |\Sigma| + 6) \rceil$ bits.

The description of M will start with $|K|$, then $|\Sigma|$ and then δ .

U uses 2-strings to simulate M .

Undecidability

Problems that have no algorithms or languages that are not recursive.

Implied by the fact that there are more languages than TMs.

HALTING Problem: Given the description of a TM M and its input x , will M halt on x ?

$$H = \{M; x : M(x) \neq \nearrow\}$$

Theorem 1. H is undecidable, i.e., not recursive.

All r.e. languages can be **reduced** to H , i.e., if we can solve HALTING we can decide **any** r.e. language.

Thus $HALTING$ is a **complete** problem for the class of r.e. languages.

HALTING Problem

Theorem 2. H is not recursive.

Proof: By contradiction.

Let M_H be a TM that decides H .

Consider another TM D that simulates M_H on input M :

$D(M) : \text{if } M_H(M; M) = \text{“yes” then } \nearrow \text{ else “yes”}.$

What is $D(D)$?

If $D(D) = \nearrow$ then $M_H(D; D) = \text{yes}$ and $D(D) \neq \nearrow$.

If $D(D) \neq \nearrow$ then $M_H(D; D) = \text{no}$ and $D(D) = \nearrow$.

Contradiction. Thus neither D nor M_H can exist.

Reduction

Let $L(M)$ be the language accepted by TM M .

Theorem 3. *The language $E = \{M : L(M) = \phi\}$ is undecidable.*

Proof: Suppose TM R decides E . We use R to construct TM S that decides H .

S does the following on input $(M; w)$:

1. Using description of M and w it constructs another TM M_1 as follows.

$M_1(x)$:

if $x \neq w$ then “no”

else run M on input w and say “yes” if M accepts.

2. Simulates R on M_1 .

3. If R says “yes” then “no” else “yes”.

Rice's Theorem

Theorem 4. *Suppose C is a proper, non-empty subset of the set of all r.e. languages. Then the following problem is undecidable: Given a TM M , is $L(M) \in C$.*

“Testing any non-trivial property of the languages recognized by TMs is undecidable.”

Proof

Assume w.l.o.g. that $\phi \notin \mathcal{C}$.

Since \mathcal{C} is nonempty, we can assume that there is a language $L \in \mathcal{C}$, accepted by machine M_L .

Let TM M_H accept H .

We will construct a machine M_x whose language is either L or ϕ .

Construct the machine M_x as follows:

$M_x(y)$: if $M_H(x) = \text{yes}$ then $M_L(y)$ else \nearrow .

We show that $L(M_x) \in \mathcal{C}$ iff $x \in H$.

If $x \in H$, then M_x on input y accepts or never halts depending on whether $y \in L$. Hence the language accepted by M_x is $L \in \mathcal{C}$.

If $x \notin H$, then $M_H(x) = \nearrow$, and M_x accepts $\phi \notin \mathcal{C}$.

A Non-R.E. Language

Theorem 5. *A language L is recursive (decidable) iff both L and L^c are r.e.*

Proof: Suppose L is recursive, then L^c is also recursive and hence both are r.e.

Assume both are r.e., accepted by TM M and M^c respectively.

Then L is decided by a TM M' which on input x simulates both M and M^c in an interleaved manner.

Since either M or M^c should accept x , L will accept or reject.

Corollary 1. *H^c is not r.e. Thus the class of r.e. languages is not closed under complement.*

Recursive Inseparability

Definition 1. Two languages are called **recursively inseparable** if there is no recursive language R such that $L_1 \cap R = \phi$ and $L_2 \subset R$.

Theorem 6. Let $L_1 = \{M : M(M) = \text{yes}\}$ and $L_2 = \{M : M(M) = \text{no}\}$. Then L_1 and L_2 are recursively inseparable.

Proof: Suppose not. Let recursive language R separate them:

$$L_1 \cap R = \phi \text{ and } L_2 \subset R.$$

Consider the TM M_R deciding R .

What is $M_R(M_R)$?

If it is “yes” then $M_R \in L_1$ and this implies $M_R(M_R) = \text{no}$.

If it is “no” then $M_R(M_R) \in L_2$ and this implies $M_R(M_R) = \text{yes}$.

Contradiction.

Corollary 2. Let $L'_1 = \{M : M(\epsilon) = \text{yes}\}$ and $L'_2 = \{M : M(\epsilon) = \text{no}\}$. Then L'_1 and L'_2 are recursively inseparable.

Proof: Suppose not. Let R' separate the two. Then we will show that we can separate L_1 and L_2 of the Theorem.

For any TM M , let M' be a TM which, on any input, generates M 's description and simulates M on it.

Consider the following TM N :

On input M :

0. If M is not a TM description accept.
1. Construct M' and test whether $M' \in R'$.
2. If yes, then accept else reject.

The recursive language decided by N separates L_1 from L_2 .