Chapter 5  
Section 4  

Undecidable Problems  
-- Halting problem revisit  
-- Reductions  
-- Problems on r.e. sets with nontrivial properties  
-- Rice’s Theorem

Universal TM

Encoding of TMs \(<M>\)
\[ K = \{<M>| M \text{ is a TM and } M \text{ halts on the input } <M>\} \]
(K is denoted by H0 in the textbook)

THM.
K is undecidable, but K is Turing-acceptable.

Let HALT = \{<M><x>| M \text{ is a TM and } M \text{ halts on the input } x\}
We could also use \(<M, x>\) to denote \(<M><x>\)
We also use H to denote HALT.

We’ll show that H is undecidable. We’re going to use reduction to prove this result.

Suppose we have two languages A and B. If there is a Turing-compatible function f (i.e., f can be computed by a TM M on all inputs & M always halts). Such that \(\forall x: x \in A \iff f(x) \in B\), written as then we say that A is reducible to B written as \(A \leq_m B\).

Proposition 1.

If \(A \leq_m B\) and B is decidable, then A must be decidable.

Proof

Assume that \(A \leq_m B\) via a reduction f then we know that f is Turing-compatible, and \(\forall x: x \in A \iff f(x) \in B\).

If B is decidable, then \(\exists\) a DTM MB that decides B. I.E., MB on any input always halts and if \(x \in B\) then MB accepts x & if \(x \not\in B\) then MB rejects x. Let Mf be a DTM that computes f. Construct a DTM MA to decide A as follows: MA(x) = MB(Mf(x)) This means that on any input x, MA first simulate Mf on x. Then MA simulates MB on the output f(x) of Mf on x.

Since Mf on any input always halts, and MB on any input always halts, we know that MA on any input will always halt. Since MB has two halting states ha and hr, Ma will also have two halting states. Now, \(\forall x\) if \(x \in A\) then f(x)\(\in B\). Hence MB accepts f(x). Since Mf(x) = f(x), we have if \(x \in A\) then MB accepts Mf(x). This means that MA accept x. Similarly, if \(x \not\in A\) then f(x)\(\not\in B\). Hence, MB rejects f(x). This means that MA rejects x. Thus, MA decides A so A is decidable.

Corollary.

If \(A \leq_m B\) and A is not decidable the B is not decidable. Now we’re ready to show that H = \{<M><x>| M is a TM & halts on x\} is not decidable by reducing H0 to H We can construct this reduction as follows:

On any instance \(<M>\) of H0, define f\(<M>\) = \(<M><M>\). Then its straightforward to see that f is Turing-computable (a machine that duplicates its input). And we have \(\forall <M>: <M> \in H0 \iff M \text{ is a TM & M halts on } <M>\) iff \(f(<M>) \in H\). Hence, \(H0 \leq_m H\) iff \(f(<M>) \in H\), thus, H is not decidable.

Proposition 2:
If $A \leq_m B$ and $B$ is Turing-acceptable, then $A$ is Turing acceptable.

**Proof**

Since $A \leq_m B$, there is a Turing-computable reduction $f$ s.t. $\forall x: x \in A$ iff $f(x) \in B$. Let $M_f$ be a DTM that computes $f$. This means that on any input $x$, $M_f$ always halts and produces $f(x)$ as its output. Assume that $B$ is Turing-acceptable. Then there is a DTM $M_B$ s.t. $B = L(M_B)$. This means that for all input $x$, if $x \in B$ then $M_B$ on $x$ halts, and if $x \notin B$ then $M_B$ on $x$ never halts. Now we construct an acceptor for $A$ as follows; $M_A(x) = M_B(M_f(x))$. On any input $x$, if $x \in A$ then $f(x) \in B$. Hence, $M_B$ on the input $f(x)$ halts. Since $f(x) = M_A(x)$, $M_B$ halts on $M_f(x)$. Thus, if $x \in A$, then $M_A$ on $x$ halts.

On the other hand, if $x \notin A$, then $f(x) \notin B$. Hence, $M_B$ on the input $f(x)$ never halts. This implies that $M_A$ on $x$ never halts. Thus $M_A$ is indeed an acceptor for $A$. So $A$ is Turing-acceptable.

**Corollary:**

If $A \leq_m B$ and $A$ is not recursively enumerable (Turing acceptable) then $B$ is not recursively enumerable. This means that we have a way to show that the language is not Turing-acceptable.

For instance, we know that $H$ is not Turing-acceptable ($H$ is Turing-acceptable but $H$ is not decidable). If we can reduce $H$ to a language $L$, the $L$ is not Turing-acceptable.

**Proposition 3:**

$A \leq_m B$ iff $A \leq_m \overline{B}$

Let $L = \{<M_1><M_2>|M_1$ and $M_2$ are TMs and $L(M_1)$ not equal $L(M_2)\}$. Then $L$ is not Turing-acceptable.

**Proof.**

Reduce $H$ to $L$ as follows:

Construct a reduction $f$ s.t. on instance (input) of $H <M><x>$, $f$ outputs two TMs $M_1$ and $M_2$. Where $M_1$ accepts everything. I.E., $L(M_1)=\Sigma^*$ and $M_2$ on input $W$ will first simulate $M$ on $x$, if $M$ halts on $x$, then $M_2$ halts on $w$. I.E., if $M$ on $x$ halts, then $L(M_2)=\Sigma^*$.

This means that $f(<M><x>)=<M_1><M_2>$ and $<M><x> \in H$ iff $L(M_1)=\Sigma^* \& L(M_2)=\Sigma^*$

iff $<M_1><M_2> \in \overline{L}$. Hence, $H \leq_m \overline{L}$. This implies that $L$ is not Turing-acceptable.

We’ll call a set of r.e. sets a property.

E.g. 

$\{\emptyset\}$ is the property of r.e. sets being empty.

$\{\Sigma^*\}$ is the property of “being full”.

$\{\text{Regular language}\}$ is the property of “being regular”

For any given property $P$, we consider languages $L_P = \{<M>|M$ is a TM & $L(M) \in P\}$.

E.g. 

$E=\{<M>|M$ is a TM & $L(M) = \emptyset\}$

$F=\{<M>|M$ is a TM & $L(M) = \Sigma^*\}$

$R=\{<M>|M$ is a TM & $L(M)$ is regular\}

All of these Languages are not decidable.

Say a property $P$ is trivial if $P$ is empty or $P$ contains all r.e. sets.
Rice’s Theorem:
For any nontrivial property P, LP is not decidable.

As a direct application of Rice’s Theorem, we know that E, F, R are not decidable.