COMPUTATION: DAY 6

BURTON ROSENBERG
UNIVERSITY OF MIAMI

Contents

1. Machines that quit 1
2. Undecidability 3
2.1. It is undecidable if a Turing machine halts 3
2.2. It is undecidable if a language is non-empty 4
2.3. It is undecidable if a language is Regular 5
2.4. Rice's Theorem 5
3. The Arithmetic Hierarchy 5
3.1. Emptiness of a language is co-RE 5
3.2. Equality of languages is neither RE nor co-RE 6

1. Machines that quit

A recursively enumerable function has two proper values, accept and reject, and a non-halting condition. An element is in the set of the machine halts accepting. The other cases the element is not in the language. In a sense, accepting means to provide proof of the element being in the set. The finite set of sets that leads to the accepting state is evidence that the element is in the set.

That an element is not in the set may or may not come with proof.

To handle this situation a bit more concretely, consider step-bounding our Turing machines. If after t steps the machine has not decided, it explicitly returns the non-halting symbol $\perp$. To accept or reject means that there exists a time bound t, and any time bound larger, in which the machine accepts or rejects.

Let $M_i(j;t)$ denote the result of the i-th Turing machine computing the j-th input for up to t logical steps. If $M_i(j)$ halts in that many steps, so does $M_i(j;t)$, with the same decision. Else $M_i(j;t) = \perp$, a symbol for having run out of time, and is therefore indecisive of the result.

Then,

$$\mathcal{L}(M_i) = \{\, j \mid \exists t \in \mathbb{N},\ M_i(j;t) = T \,\}$$

Theorem 1.1. If a language $A \subseteq \Sigma^*$ and its complement $\overline{A}$ are both recursively enumerable, then A is recursive. (So is $\overline{A}$.)

Proof: There exists time bounded TM M_r and M_s whose languages are A and $\overline{A}$, respectively. Enumerate the set $\{ r, s \} \times \mathbb{N}$ as follows,

$$\langle\, (k,t)\, \rangle = \langle\, (r,0), (s,0), (r,1), (s,1), (r,2), \ldots \rangle$$

and create an interleaving computation as a machine,

$$M'(j,(k,t)) = M_k(j;t).$$

If $j \in A$ then there is a t for which $M_s(j;t) = T$; and otherwise there is a t for which $M_r(j;t) = T$. So M' always halts and its accepting set is A and its rejecting set is $\overline{A}$. □

Theorem 1.2. The language of non-empty languages,

$$\overline{E_{TM}} = \{\, i \in \mathbb{N} \,|\, \mathcal{L}(M_i) \neq \emptyset \,\}$$

is recursively enumerable.

Proof: Consider the enumeration,

$$d = \langle\, (0,0), (1,0), (0,1), (2,0), (1,1), (0,2), (3,0), \ldots \rangle.$$

This is a sequence of pairs that enumerates over all pairs of integers such that any pair is a finite distance along the sequence. A program for this enumeration can be written,

```
def d(n):
    i, d = 0, 0
    while True:
        for j in range(i+1):
            if d==n: return (i-j,j)
            d += 1
        i += 1
```

If M_i accepts something, it does so on a specific j in a special time bound t. Enumerating that (j,t) will be found.

$$i \in \overline{E_{TM}} \iff \exists\, n \text{ such that } d(n) = (j,t) \text{ and } M_i(j;t) = T.$$

This describes a machine accepting exactly the indices of TM's whose languages are non-empty. □

Note the formula for the machine has a single existential quantifier over the natural numbers. This is the basic format when describing a recursively enumerable set; as it is the format for finding an accepting computation.

2. Undecidability

Definition 2.1. A *recursive function* is and function $f : \Sigma^* \to \Sigma^*$ such that there exists an always-halting Turing machine M that computes the function in the following sense: When M is started with string s on its tape then it runs and halts with string $f(s)$ in its tape.

Definition 2.2. Given $A, B \subseteq \Sigma^*$, a *reduction of A to B* is a recursive function $f : \Sigma^* \to \Sigma^*$ that preserves the set inclusion,

(1) $\forall a \in \implies f(a) \in B$,
(2) $\forall a \notin A \implies f(a) \notin B$.

A reduction from A to B is denoted $A \leq_m B$.

Theorem 2.1. If $A \leq_m B$ and B is a recursive set, then A is a recursive set. Therefore, if B is not recursive, A is not recursive.

Proof: Let f be the reduction map. If B is recursive, then there as a recursive function $g : \Sigma^* \to \{T, F\}$. The function $g \circ f$ is recursive. Consider the Turing machine that computes this function and accepts if T is left on the tape and rejects of F is left on the tape. This Turing machine decides the set A.

Theorem 2.2. If $A \leq_m B$ and B is a recursively enumerable set, then A is a recursively enumerable set. Therefore if A is not recursively enumerable, B is not recursively enumerable.

Proof: Similar to the above proof.

2.1. **It is undecidable if a Turing machine halts.**

Define the halting problem for turing machines, H_{TM},

$$H_{TM} = \{\, (i, j) \in \mathbb{N} \times \mathbb{N} \,|\, M_i(j) \neq \perp \,\}$$

Consider the map $(i, j) \mapsto (f(i), j)$ which will act as,

$$M_{f(i)}(j) \equiv \text{if } M_i(j) \text{ then } T \text{ else } \perp$$

The function f is a recursive function which which retrieves the machine definition of machine i then operates on the definition to replace any transitions to a reject state to a transition to an infinite loop. It then gets the index $f(i)$ for this machine and outputs it on its tape, along with a copy of the original j.

It is a reduction,

$$\begin{array}{rcl}
(i,j)\in A_{TM} & \Longrightarrow & M_i(j)=T \\
 & \Longrightarrow & M_{f(i)}(j)=T \\
 & \Longrightarrow & (f(i),j)\in H_{TM}. \\
(i,j)\notin A_{TM} & \Longrightarrow & M_i(j)\neq T \\
 & \Longrightarrow & M_{f(i)}(j)=\perp \\
 & \Longrightarrow & (f(i),j)\notin H_{TM}.
\end{array}$$

This is the reduction

$$A_{TM}\leq H_{TM}$$

and since A_{TM} is not recursive then H_{TM} is not recursive.

2.2. **It is undecidable if a language is non-empty.** Consider the map

$$(i,j)\mapsto (f(i),j)$$

which will act as,

$$M_{f(i,j)}(k)\equiv \text{if } M_i(j) \text{ then } T \text{ else } \perp$$

The function f is a recursive function which which retrieves the machine definition of machine i then operates on the definition to replace any transitions to a reject state to a transition to an infinite loop. It then adds states to write the given j onto the tape followed by a state transition to the start of the modified version of M_i. The function then index $f(i,j)$ for this machine and writes it to the tape.

$$\begin{array}{rcl}
(i,j)\in A_{TM} & \Longrightarrow & M_i(j)=T \\
 & \Longrightarrow & \forall k\ M_{f(i,j)}(k)=T \\
 & \Longrightarrow & \mathcal{L}(M_{f(i,j)})=\Sigma^* \\
 & \Longrightarrow & f(i,j)\in\overline{E_{TM}}. \\
(i,j)\notin A_{TM} & \Longrightarrow & M_i(j)\neq T \\
 & \Longrightarrow & \forall k\ M_{f(i,j)}(k)=\perp \\
 & \Longrightarrow & \mathcal{L}(M_{f(i,j)})=\emptyset \\
 & \Longrightarrow & f(i,j)\notin\overline{E_{TM}}
\end{array}$$

This is the reduction

$$A_{TM}\leq_m \overline{E_{TM}}$$

and since A_{TM} is not recursive then $\overline{E_{TM}}$ is not recursive. But $\overline{E_{TM}}$ is recursively enumerable, therefore E_{TM} is not recursively enumerable. Otherwise said, in general it is unprovable that any given Turing Machine accepts no strings.

2.3. **It is undecidable if a language is Regular.**

Consider the TM N such that

$$\mathcal{L}(N) = \{\, 0^i\, 1^i \,\}$$

Then,

$$M_{f(i,j)}(k) \equiv \text{if } M_i(j) \text{ then } N(k) \text{ else } \perp$$

accepts a non-regular language if $M_i(j) = T$ and a the regular language $\emptyset$ else. This is reduction from the halting problem to recognizing regular languages. Hence recognizing regular languages is undecidable.

2.4. **Rice's Theorem.**

Theorem 2.3. Any non-trivial property of recursively enumerable sets is undecidable.

Proof: The above pattern of reductions is generalized. If the property P is non-trivial, meaning all languages have or do not have the property, let the property be defined in such a way that the empty language does not have the property, $\emptyset \not\in P$, and the language of machine M_j has the property, $\mathcal{L}(M_j) \in P$. Let $P(j)$ be the machine the recognizes the property. Then the construction,

$$M_{f(i,j)}(k) \equiv \text{if } M_i(j) \text{ then } P(k) \text{ else } \perp$$

is a reduction $A_{TM} \leq_m P$. Therefore P is undecidable.

3. The Arithmetic Hierarchy

The term *undecidable* refers to any non-recursive set. So far we have found two such sets, those that are *recursively enumerable* and those that are *co-recursively enumerable*, complements of recursively enumerable sets. We have employed reductions from a recursively enumerable set to a unknown set to show it is undecidable, and then given a recognizer for said unknown set to show it is recursively enumerable.

3.1. **Emptiness of a language is co-RE.**

The set $\neg E_{TM}$ was shown to be undecidable. It was also shown to be recursively enumerable. There for its complement E_{TM} is co-recursively enumerable. It cannot be decided, recognized, or enumerated.

3.2. **Equality of languages is neither RE nor co-RE.** The question of whether two Turing machines differ require a more complicated statement that alternates quantifiers,

$$\overline{EQ_{TM}} = \{\, (i,j) \in \mathbb{N} \times \mathbb{M} \,|\, \exists (k, t_k)\ \forall t \geq t_k, M_i(k;t) \neq M_j(k;t)\,\}$$

The case here is when the difference between the two functions is a single point where M_i halts and M_j does not. On one level, we have definite evidence, that the disagreement was on input k, but also indefinite evidence, that for one machine a particular step count t suffices, but for the other machine we have to consider all step counts.

Note that a k specific lower bound for t is needed on the universal quantifier, $t > t_k$, since for small t both machines could still be computing and hence equal at that (k,t) point.

Consider the map $h_1(i,j) \mapsto (f(i,j), M_\emptyset)$ where,

$$M_{f(i,j)}(k) \equiv \text{if } M_i(j) \text{ then } T \text{ else } \bot$$

and $M_\emptyset$ is any Turing machine that accepts nothing. Previously it was described how f is a recursive function. It preserves truth from acceptance to non-equality.

$$\begin{array}{rcl} (i,j) \in A_{TM} & \Longrightarrow & M_i(j) = T \\ & \Longrightarrow & \mathcal{L}(M_{f(i,j)}) = \Sigma^* \\ & \Longrightarrow & M_{f(i,j)} \neq M_\emptyset \\ & \Longrightarrow & (f(i,j), M_\emptyset) \in \neg EQ_{TM}. \\ (i,j) \notin A_{TM} & \Longrightarrow & M_i(j) \neq T \\ & \Longrightarrow & \mathcal{L}(M_{f(i,j)} = \emptyset \\ & \Longrightarrow & M_{f(i,j)} = M_\emptyset \\ & \Longrightarrow & (f(i,j), M_\emptyset) \notin \neg EQ_{TM} \end{array}$$

Consider the very similar map $h_2(i,j) \mapsto (f(i,j), M_{\Sigma^*})$ where M_{Σ^*} is the machine that accepts everything. It preserves truth from acceptance to equality,

$$\begin{array}{rcl} (i,j) \in A_{TM} & \Longrightarrow & M_i(j) = T \\ & \Longrightarrow & (f(i,j), M_{\Sigma^*}) \in EQ_{TM}. \\ (i,j) \notin A_{TM} & \Longrightarrow & M_i(j) \neq T \\ & \Longrightarrow & (f(i,j), M_{\Sigma^*}) \notin EQ_{TM} \end{array}$$

So we have two reductions,

$$A_{TM} \leq_m \neg EQ_{TM}, \;\; A_{TM} \leq_m EQ_{TM}$$

so neither can be recursive. As reductions are stable by complementing both sides, we also have,

$$\neg A_{TM} \leq_m EQ_{TM}, \;\; \neg A_{TM} \leq_m \neg EQ_{TM}$$

So neither can be recursively enumerable either.

These are sets that cannot be decided, and further neither the set nor its complement can be recognized.