1 Rice's Theorem

1.1 Properties

Checking Properties

Given M

$$\left.\begin{array}{l}\text{Does } \mathbf{L}(M) \text{ contain } \langle M\rangle?\\ \text{Is } \mathbf{L}(M) \text{ non-empty?}\\ \text{Is } \mathbf{L}(M) \text{ empty?}\end{array}\right\}\text{Undecidable}$$

$$\left.\begin{array}{l}\text{Is } \mathbf{L}(M) \text{ infinite?}\\ \text{Is } \mathbf{L}(M) \text{ finite?}\\ \text{Is } \mathbf{L}(M) \text{ co-finite (i.e., is } \overline{\mathbf{L}(M)} \text{ finite)?}\\ \text{Is } \mathbf{L}(M) = \Sigma^*?\end{array}\right\}\text{Undecidable}$$

None of these properties can be decided. This is the content of Rice's Theorem.

Properties

Definition 1. A property of languages is simply a set of languages. We say L *satisfies* the property $\mathbb{P}$ if $L \in \mathbb{P}$.

Definition 2. For any property $\mathbb{P}$, define language $L_{\mathbb{P}}$ to consist of Turing Machines which accept a language in $\mathbb{P}$:

$$L_{\mathbb{P}} = \{\langle M\rangle \mid \mathbf{L}(M) \in \mathbb{P}\}$$

Deciding $L_{\mathbb{P}}$: deciding if a language represented as a TM satisfies the property $\mathbb{P}$.

- *Example:* $\{\langle M\rangle \mid \mathbf{L}(M) \text{ is infinite}\}$; $E_{\text{TM}} = \{\langle M\rangle \mid \mathbf{L}(M) = \emptyset\}$
- *Non-example:* $\{\langle M\rangle \mid M \text{ has 15 states}\}$ $\longleftarrow$ This is a property of TMs, and not languages!

Trivial Properties

Definition 3. A property is trivial if either it is not satisfied by any r.e. language, or if it is satisfied by all r.e. languages. Otherwise it is non-trivial.

Example 4. Some trivial properties:

- $\mathbb{P}_{\text{ALL}}$ = set of all languages
- $\mathbb{P}_{\text{R.E.}}$ = set of all r.e. languages
- $\overline{\mathbb{P}}$ where $\mathbb{P}$ is trivial
- $\mathbb{P} = \{L \mid L \text{ is recognized by a TM with an even number of states}\} = \mathbb{P}_{\text{R.E.}}$

Observation. For any trivial property $\mathbb{P}$, $L_{\mathbb{P}}$ is decidable. (Why?) Then $L_{\mathbb{P}} = \Sigma^*$ or $L_{\mathbb{P}} = \emptyset$.

1.2 Main Theorem

Rice's Theorem

Proposition 5. *If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.*

- Thus $\{\langle M\rangle \mid \mathbf{L}(M) \in \mathbb{P}\}$ is not decidable (unless $\mathbb{P}$ is trivial)

We cannot algorithmically determine any interesting property of languages represented as Turing Machines!

Properties of TMs

Note. Properties of TMs, as opposed to those of languages they accept, may or may not be decidable.

Example 6.

$$\left.\begin{array}{l}\{\langle M\rangle \mid M \text{ has 193 states}\} \\ \{\langle M\rangle \mid M \text{ uses at most 32 tape cells on blank input}\}\end{array}\right\} \text{Decidable}$$

$$\left.\begin{array}{l}\{\langle M\rangle \mid M \text{ halts on blank input}\} \\ \{\langle M\rangle \mid \text{ on input 0011 } M \text{ at some point writes the} \\ \qquad\qquad \text{symbol \$ on its tape}\}\end{array}\right\} \text{Undecidable}$$

Proof of Rice's Theorem

Rice's Theorem
If $\mathbb{P}$ is a non-trivial property, then $L_{\mathbb{P}}$ is undecidable.

Proof. Suppose $\mathbb{P}$ non-trivial and $\emptyset \notin \mathbb{P}$. If $\emptyset \in \mathbb{P}$, then in the following we will be showing $L_{\overline{\mathbb{P}}}$ is undecidable. Then $L_{\mathbb{P}} = \overline{L_{\overline{\mathbb{P}}}}$ is also undecidable.

Recall $L_{\mathbb{P}} = \{\langle M\rangle \mid \mathbf{L}(M) \text{ satisfies } \mathbb{P}\}$. We'll reduce A_{TM} to $L_{\mathbb{P}}$. Then, since A_{TM} is undecidable, $L_{\mathbb{P}}$ is also undecidable. Broadly the idea behind the reduction is as follows. Since $\mathbb{P}$ is non-trivial, at least one r.e. language satisfies $\mathbb{P}$. i.e., $\mathbf{L}(M_0) \in \mathbb{P}$ for some TM M_0. We will show a reduction f that maps an instance $\langle M, w\rangle$ for A_{TM}, to N such that

- If M accepts w then N accepts the same language as M_0. Then $\mathbf{L}(M) = \mathbf{L}(M_0) \in \mathbb{P}$
- If M does not accept w then N accepts $\emptyset$. Then $L(N) = \emptyset \notin \mathbb{P}$

Thus, $\langle M, w\rangle \in A_{\text{TM}}$ iff $N \in L_{\mathbb{P}}$.

We now describe the reduction precisely. The reduction f maps $\langle M, w\rangle$ to $\langle N\rangle$, where N is a TM that behaves as follows:

```
On input x
    Ignore the input and run M on w
    If M does not accept (or doesn't halt)
        then do not accept x (or do not halt)
    If M does accept w
        then run M0 on x and accept x iff M0 does.
```

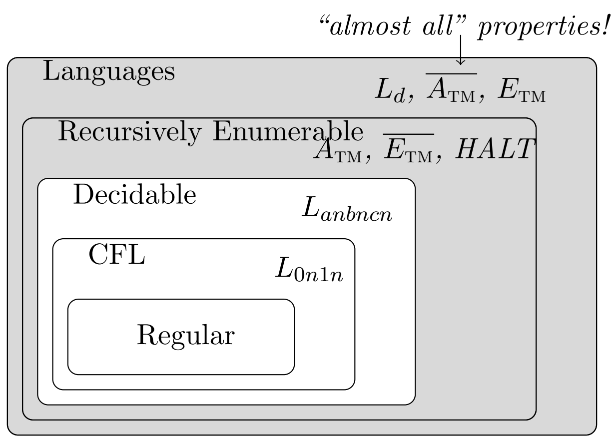
Notice that indeed if M accepts w then $\mathbf{L}(N) = \mathbf{L}(M_0)$. Otherwise $\mathbf{L}(N) = \emptyset$. □

Rice's Theorem
Recap

Every non-trivial property of r.e. languages is undecidable

- Rice's theorem says nothing about properties of Turing machines
- Rice's theorem says nothing about whether a property of languages is recurisvely enumerable or not.

Big Picture . . . again



---

2 Closure Properties

2.1 Decidable Languages

Boolean Operators

Proposition 7. *Decidable languages are closed under union, intersection, and complementation.*

Proof. Given TMs M_1, M_2 that decide languages L_1, and L_2

- A TM that decides $L_1 \cup L_2$: on input x, run M_1 and M_2 on x, and accept iff either accepts. (Similarly for intersection.)
- A TM that decides $\overline{L_1}$: On input x, run M_1 on x, and accept if M_1 rejects, and reject if M_1 accepts. □

Regular Operators

Proposition 8. *Decidable languages are closed under concatenation and Kleene Closure.*

Proof. Given TMs M_1 and M_2 that decide languages L_1 and L_2.

- A TM to decide L_1L_2: On input x, for each of the $|x|+1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to decide L_1^*: On input x, if $x = \epsilon$ accept. Else, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \dots w_k$ ($w_i \neq \epsilon$): run M_1 on each w_i and accept if M_1 accepts all. Else reject. □

Inverse Homomorphisms

Proposition 9. *Decidable languages are closed under inverse homomorphisms.*

Proof. Given TM M_1 that decides L_1, a TM to decide $h^{-1}(L_1)$ is: On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts. □

Homomorphisms

Proposition 10. *Decidable languages are not closed under homomorphism*

Proof. We will show a decidable language L and a homomorphism h such that $h(L)$ is undecidable

- Let $L = \{xy \mid x \in \{0,1\}^*, y \in \{a,b\}^*, x = \langle M, w\rangle$, and y encodes an integer n such that the TM M on input w will halt in n steps $\}$
- L is decidable: can simply simulate M on input w for n steps
- Consider homomorphism h: $h(0) = 0$, $h(1) = 1$, $h(a) = h(b) = \epsilon$.
- $h(L) =$ HALT which is undecidable. □

2.2 Recursively Enumerable Languages

Boolean Operators

Proposition 11. *R.E. languages are closed under union, and intersection.*

Proof. Given TMs M_1, M_2 that recognize languages L_1, L_2

- A TM that recognizes $L_1 \cup L_2$: on input x, run M_1 and M_2 on x *in parallel*, and accept iff either accepts. (Similarly for intersection; but no need for parallel simulation) □

Complementation

Proposition 12. *R.E. languages are not closed under complementation.*

Proof. A_{TM} is r.e. but $\overline{A_{\text{TM}}}$ is not. □

Regular Operations

Proposition 13. *R.E languages are closed under concatenation and Kleene closure.*

Proof. Given TMs M_1 and M_2 recognizing L_1 and L_2

- A TM to recognize $L_1 L_2$: On input x, do *in parallel*, for each of the $|x| + 1$ ways to divide x as yz: run M_1 on y and M_2 on z, and accept if both accept. Else reject.
- A TM to recognize L_1^*: On input x, if $x = \epsilon$ accept. Else, do *in parallel*, for each of the $2^{|x|-1}$ ways to divide x as $w_1 \ldots w_k$ ($w_i \neq \epsilon$): run M_1 on each w_i and accept if M_1 accepts all. Else reject. □

Homomorphisms

Proposition 14. *R.E. languages are closed under both inverse homomorphisms and homomorphisms.*

Proof. Let TM M_1 recognize L_1.

- A TM to recognize $h^{-1}(L_1)$:On input x, compute $h(x)$ and run M_1 on $h(x)$; accept iff M_1 accepts.
- A TM to recognize $h(L_1)$: On input x, start going through all strings w, and if $h(w) = x$, start executing M_1 on w, using *dovetailing* to interleave with other executions of M_1. Accept if any of the executions accepts. □
