

Rice's theorem

- Some of the undecidable languages we've seen:

$$\text{EMPTY} = \{\langle M \rangle \mid L(M) = \emptyset\}$$

$$\text{FINITE} = \{\langle M \rangle \mid L(M) \text{ is finite}\}$$

$$\text{REGULAR} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$$

- What do these languages have in common? They all ask about the language of the input TM.
- Rice's theorem says that all non-trivial languages of this type are undecidable.
- **Definition:** Let \mathcal{R} be the set of all recognizable languages. A *property* of recognizable languages is a subset $P \subseteq \mathcal{R}$. A property P is *trivial* if $P = \emptyset$ or $P = \mathcal{R}$, otherwise it is non-trivial. The language L_P corresponding to a property P is:

$$L_P = \{\langle M \rangle \mid L(M) \in P\}$$

- **Theorem:** Let P be a non-trivial property of recognizable languages. Then L_P is undecidable.
- **Proof:** Consider two cases, $\emptyset \notin P$ and $\emptyset \in P$.

– *Case 1:* $\emptyset \notin P$. Show $A_{TM} \leq_m L_P$. Since P is non-trivial, there is a language $L_1 \in P$ and a language $L_2 \in \mathcal{R} \setminus P$. Since $P \subseteq \mathcal{R}$, both L_1 and L_2 are recognizable, so there exist TMs M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. Define $f(\langle M, w \rangle) = \langle M' \rangle$, where M' = "on input x ...

1. Run M on input w
2. If M accepts w then run M_1 on input x and accept iff M_1 accepts
3. If M rejects w then run M_2 on input x and accept iff M_2 accepts

We need to show that $\langle M, w \rangle \in A_{TM} \iff L(M') \in P$.

(\Rightarrow) If $\langle M, w \rangle \in A_{TM}$ then M accepts w , so $L(M') = L(M_1) = L_1 \in P$.

(\Leftarrow) If $\langle M, w \rangle \notin A_{TM}$ then either M rejects w , or M doesn't halt on w . If M rejects w then $L(M') = L(M_2) = L_2 \notin P$. If M doesn't halt on w then $L(M') = \emptyset \notin P$ (remember, we assumed $\emptyset \notin P$ for this case).

– *Case 2:* $\emptyset \in P$. Let $\bar{P} = \mathcal{R} \setminus P$ be the complement property of P . Then $\emptyset \notin \bar{P}$, so by case 1 $L_{\bar{P}}$ is undecidable. But,

$$L_{\bar{P}} = \{\langle M \rangle \mid L(M) \notin P\} = \overline{L_P}$$

Therefore $\overline{L_P}$ is not decidable, and since the complement of an undecidable language is undecidable, it follows that L_P is undecidable.

Some interesting undecidable languages

Debugging/Verification

- The halting problem is related to debugging: it asks whether a program halts on a given input.

- Even more applicable is the following language:

$$\text{ALWAYS-HALTS} = \{\langle M \rangle \mid M \text{ halts on every input}\}$$

- This language is undecidable (try reducing from A_{TM}). Thus even the simplest task of a programmer, ensuring that the program never enters an infinite loop, cannot be effectively automated in the general case.
- Let B be a language (the “target” language). An undecidable language related to verification is:

$$L_B = \{\langle M \rangle \mid L(M) = B\}$$

- The idea is that you’re trying to write a program that decides B , and using a decider for L_B you could check that your program is correct; but a simple application of Rice’s theorem shows that this language is undecidable.

Ambiguity of context-free grammars

- Recall that a context-free grammar (CFG) consists of rules like:

$$S \rightarrow AB$$

$$A \rightarrow xy$$

$$A \rightarrow yx$$

$$A \rightarrow AA$$

$$B \rightarrow x$$

$$B \rightarrow y$$

$$B \rightarrow BB$$

- A CFG is ambiguous if there exists a string that has two different derivations; in the above example, the string xyx can be derived as

$$S \rightarrow AB \rightarrow AAB \rightarrow xyAB \rightarrow xyxyB \rightarrow xyxyx$$

$$S \rightarrow AB \rightarrow ABB \rightarrow AB BB \rightarrow xyBBB \rightarrow xyxBB \rightarrow xyxyB \rightarrow xyxyx$$

- CFGs are widely used, for example in compiler design. A useful property for a CFG is that it is unambiguous; however, the language

$$\text{CFG-AMB} = \{\langle G \rangle \mid G \text{ is an ambiguous CFG}\}$$

is undecidable.

Number theory

- A first order arithmetic formula consists of
 - Variables: x_1, x_2, x_3, \dots
 - Functions: $+$ and \times
 - Predicates: $=$
 - Connectives: $\wedge, \vee, \neg, \rightarrow, \dots$
 - Constants: $0, 1, 2, \dots$
 - Quantifiers: \forall, \exists
- A formula is interpreted over \mathbb{N} , i.e. the variables can take on values in \mathbb{N} and $+, \times, =$ have their usual meaning.
- Some examples:

$$\forall x \exists y \exists z (z \neq 0 \wedge y = x + z)$$

(“there is no largest number”)

$$\exists x \forall y \forall z (x = y \times z \implies (y = 1 \vee z = 1))$$

(“there exists a prime number”)

$$\forall w \exists x \exists y \forall z_1 \forall z_2 (x = w + y \wedge (x = z_1 \times z_2 \implies (z_1 = 1 \vee z_2 = 1)))$$

(“there is no largest prime number”)

- It would be extremely useful to have an algorithm that would decide whether a particular statement in the above form is true; however,

$$L_{\text{arithmetic}} = \{\langle f \rangle \mid f \text{ is a first-order formula that is true over } \mathbb{N}\}$$

is undecidable.

Post's correspondence problem

- This problem can be thought of as a game: given a finite number of types of “dominoes”, where each domino has a string written on the top, and a string written on the bottom, does there exist a sequence of these dominoes (with no limit on the number of times a particular type can be used), such that the concatenation of the top strings equals the concatenation of the bottom strings?
- For example, if the types of dominoes are:

$$\begin{pmatrix} 10 \\ 0110 \end{pmatrix}, \begin{pmatrix} 1101 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 00 \end{pmatrix}, \begin{pmatrix} 01 \\ 11 \end{pmatrix}, \begin{pmatrix} 01 \\ 1 \end{pmatrix}$$

Then a matching sequence would be:

$$\begin{pmatrix} 0 \\ 00 \end{pmatrix} \begin{pmatrix} 01 \\ 11 \end{pmatrix} \begin{pmatrix} 1101 \\ 1 \end{pmatrix} \begin{pmatrix} 10 \\ 0110 \end{pmatrix}$$

- This problem is undecidable. Next class we will see a reduction from A_{TM} to Post's correspondence problem.