Two more languages that are not recursively enumerable

Two complementary languages

- Consider these two languages:
  \[ L_r = \{ \langle M \rangle : L(M) \text{ is recursive} \} = \overline{L_{nr}} \]
  \[ L_{nr} = \{ \langle M \rangle : L(M) \text{ is not recursive} \} \]

- The textbook shows that both \( L_r \) and \( L_{nr} \) are NOT rec. enum.
- Both proofs involve building “monster” machines
- Both proofs involve showing that if the new language were rec. enum. then \( \overline{L_u} \) would be rec. enum.

Cannot happen

- Now, \( L_u \) is rec. enum (use a universal TM) so if \( L_u \) were rec. enum, then both \( L_u \) and \( \overline{L_u} \) would be recursive (Theorem 8.3)
- However, \( L_u \) is not recursive
First, $L_r$

- Suppose $L_r$ is rec. enum. and $M_r$ is a TM for $L_r$. Let $M_u$ be a universal TM.
- Use an algorithm to convert an input $<M, w>$ into a machine $M'$ such that
  - $L(M') = \emptyset$ if $<M, w> \notin L_u$
  - $L(M') = L_u$ if $<M, w> \in L_u$

**Details**

- The construction of $M'$ from $<M, w>$ and $M_u$ is basically a matter of
  - putting an extra tape to simulate $M$ on $w$ and
  - a transition from the final state of $M$ to the initial state of $M_u$
- It is obvious that
  - $L(M') = \emptyset$ if $<M, w> \notin L_u$
  - $L(M') = L_u$ if $<M, w> \in L_u$

**Next, the monster**

- We convert $M'$ to $M''$ of the Chapter 8 form and obtain its binary encoding $<M''>$
- Feed $<M''>$ to $M_r$
the conclusion

- $M_r$ accepts $<M''>$ if and only if
- $L(M') = L(M'')$ is recursive if and only if
- $L(M') = \emptyset$, since $L_u$ is not recursive if and only if
- $<M, w> \notin L_u$
- The combination is a TM for $\overline{L_u}$
- Impossible, so $M_r$ does not exist

Next we look at $L_{nr}$

- Suppose $L_{nr}$ is rec. enum. and $M_{nr}$ is a TM for $L_{nr}$. Let $M_u$ be as before
- Use an algorithm to convert an input $<M, w>$ into a machine $M'$ such that
  - $L(M') = \Sigma^*$ if $<M, w> \in L_u$
  - $L(M') = L_u$ if $<M, w> \notin L_u$
Use $M_{nr}$

- We convert $M'$ to $M''$ of the Chapter 8 form and obtain its binary encoding $<M''>$
- Feed $<M''>$ to $M_{nr}$

The conclusion

- $M_{nr}$ accepts $<M''>$ if and only if $L(M') = L(M'')$ is not recursive if and only if $L(M') = L_u$ since $\Sigma^*$ is recursive if and only if $<M,w> \notin L_u$
- The combination is a TM for $\overline{L_u}$
- Impossible, so $M_{nr}$ does not exist

Picture

Two more dots in the diagram
What was not proved

- \( L_r = \{ <M> : L(M) \text{ is recursive} \} \) is NOT \( L_h = \{ <M> : M \text{ halts on all inputs} \} \)
- It is normal for \( L = L(M) \) to be recursive, when some of the machines \( M \) for the language fail to halt on strings that are not in \( L \)
- It is another issue whether \( L_h \) is rec. enum. or recursive

Turing Machines

Decidable and recursively enumerable properties

Properties

- We need Rice’s theorem to make progress on this problem
- We look at properties of recursively enumerable languages
- Let \( \mathcal{S} \) be a set of rec. enum. languages with a common definable property

Examples

- For example:
  - \( S_1 = \{ L : L \text{ is recursive} \} \)
  - \( S_2 = \{ L : L \text{ is not recursive} \} \)
  - \( S_3 = \{ L : L = \emptyset \} = \{ \emptyset \} \)
  - \( S_4 = \{ L : L = (0+1)^* \} = \{(0+1)^* \} \)
  - \( S_5 = \{ L : L \text{ is finite} \} \)
  - \( S_6 = \{ L : L \text{ is infinite} \} \)
More examples

- \( S_7 = \{ L : L \subseteq 0^* \} \)
- \( S_8 = \{ L : L \cap 0^* \neq \emptyset \} \)
- \( S_9 = \{ L : L \cap 0^* = \emptyset \} \)
- \( S_{10} = \{ L : L \cap L_u \neq \emptyset \} \)
- \( S_{11} = \{ L : L \cap L_u = \emptyset \} \)
- \( S_{12} = \{ L : L \text{ has 26 elements} \} \)
- \( S_{13} = \{ L : L \text{ is regular} \} \)
- \( S_{14} = \{ L : L \text{ is a CFL} \} \)

Properties are sets

- We actually call \( S \) a property of rec. enum. languages, e.g. if
- \( S_{13} = \{ L : L \text{ is regular} \} \)
- we say \( S_{13} \) is the property that \( L \) be regular

Corresponding TM's

- For each property \( S \), we associate a language of Chapter 8-style TM encodings:
  - \( L_S = \{ <M> : L(M) \text{ is in } S \} \)
    = \{ <M> : L(M) \text{ has property } S \} \)
- We say that a property \( S \) (of rec. enum. languages) is decidable if \( L_S \) is recursive and undecidable otherwise:

Why \( L_S \)?

- It would be hopeless to try to apply an algorithm to an infinite entity like a typical language \( L \)
- instead we try to find an algorithm that can be applied to a finite entity such as the finite set of transitions of the TM for \( L \)
Trivial properties (all or nothing)

- The property $S$ is *trivial* if and only if $S$ is one of the following two sets:
  - $S = \emptyset$
  - $S = \text{the set of all rec. enum. languages}$
    $= \{ L \subseteq (0+1)^* : L \text{ is recognized by a TM} \}$

Rice’s theorem

- Rice’s Theorem
  The property $S$ is decidable if and only if it is trivial
- We have been practicing the method of proof

Proof

- We want to prove
  $L_S = \{ <M> : L(M) \text{ has property } S \}$
  is not recursive.
- If $L_S = \{ <M> : L(M) \text{ has property } S \}$
  were recursive then
  $L_S = \{ <M> : L(M) \text{ doesn’t have prop. } S \}$
  would be recursive

Assume $\emptyset$ does not have property $S$

- If $\emptyset$ had property $S$, then we would work on proving $L_S$ is not recursive
- Since the problems are the same, we will simply assume $\emptyset$ does not have property $S$, and prove $L_S$ cannot be recursive in this case
We are assuming $S$ is not empty

- First, assume $S$ is not empty
- Take $L$ in $S$ (and note that $L \neq \emptyset$)
- Since everything in $S$ is rec. enum., we can take a TM $M_L$ for $L$
- Suppose $S$ is decidable and the halting TM $M_S$ accepts $L_S$

Another $M'$

- Given $<M, w>$, build $M'$ so that $L(M')$ has property $S$ if and only if $M$ accepts $w$
- Given $<M, w>$ and $M_L$, we can add some transitions to create $M'$

The new machine

- $M'$ first runs $M$ on $w$-- this may not halt
- IF $M$ halts and ACCEPTS $w$, then $M'$ runs $M_L$ on $x$
- $M'$ accepts $x$ if $M_L$ accepts $x$

The contradiction

- Thus
- $L(M') = L$ if $<M, w> \in L_u$
- $L(M') = \emptyset$ if $<M, w> \notin L_u$
- We convert $M'$ to a Chap. 8 TM $M''$ and run $M_S$ on $<M''>$
The halting monster machine

Since $\emptyset$ does not have the property, we only accept $<M''>$ if $L(M') = L$, i.e. if and only if $<M,w> \in L_u$

We have an algorithm for $L_u$ (which does not exist)

We have proved that if $S$ is a non-trivial property, then it cannot be decidable

It is easy to see that if $S$ is a trivial property, then it must be decidable

Conclusion

All the properties $S_1$ through $S_{14}$ are non-trivial and therefore undecidable

Notice that these are not theorems about properties of TM's directly, they are properties of the languages the TM's accept

two example machines

Build 2 TM's, one that accepts all machine encodings and one that accepts nothing (very easy)

These TM's accept the languages $L_S = \{<M> : L(M) \text{ is rec. enum.}\}$

$= 1(0+1)^*+0$ and $L_S = \emptyset$
The halting problem for TM's is undecidable

- We go through the solution to problem 8.3 of the textbook, given at the end of the exercises
- \( L_h = \{<M> : M \text{ halts on all inputs} \} \) is not recursive
- We prove that if \( L_h \) is recursive then a non-trivial property would be recursive

We need an \( M' \)

- Given \(<M>\), construct \( M' \) so that \( L(M) = L(M') \) and \( M' \) ONLY halts on \( L(M) \)
- \( M' \) shifts its input one space to the right, putting “$” at the first position

What to do at the left-hand end

- Move back to the start of \( w \) and simulate \( M \) on \( w \)
- If \( M \) reaches “$”, it would have halted and failed on \( w \); for every \( q \) in \( M \) add a transition \( \delta(q, $) = (p_1, $, R) \), for a new state \( p_1 \)
What to do if $M$ halts

- In $M'$, add transition $\delta(q,X) = (p_1,X,R)$, whenever $q$ is not final and $\delta(q,X)$ is undefined in $M$
- If $M$ halts (and fails) in a state $q$ with $X$ at the read/write head, $M'$ will move to $p_1$

send $M'$ into an infinite loop

- To force $M'$ to run for ever, add transitions $\delta(p_1,X) = (p_2,X,L)$ for all $X$
- Then add transitions $\delta(p_2,X) = (p_1,X,R)$ for all $X$
- Once we get into $p_1$ and $p_2$ we run for ever

$M'$ has the correct properties

- Of course if $M$ ran for ever, then $M'$ runs for ever
- $M'$ accepts $w$ if and only if $M$ accepts $w$ and $M'$ accepts $w$ if and only if $M'$ halts on $w$

A non-trivial property would be decidable

- Suppose $L_h$ is recursive and $M_h$ is the halting TM for $L_h$
- Construct an algorithm for $L_{S_4} = \{<M>: L(M) = (0+1)^* \}$ as follows:
Contradiction

- Given $<M>$, construct $M'$ as before and create a Chapter 8 encoding $<M''>$ to feed to $M_h$
- If $M_h$ accepts $<M''>$
- then $M'$ halts on all inputs and accepts all inputs, which happens if and only if
  - $L(M) = (0+1)^*$, since $L(M) = L(M')$

apply Rice's theorem

- All together we get an algorithm for the set of TM encodings, for which the property is the non-trivial property $L = (0+1)^*$
- Such an algorithm does not exist by Rice’s theorem

Similar results

- Suppose we take the problem “Does $M$ halt on a fixed $w$?”
- Consider the property $S = \{ L : w \in L \}$
- We know that $S$ is undecidable because it is non-trivial, i.e. $\{<M> : w \in L(M) \}$ is not recursive

halting on a fixed string

- Suppose we had an algorithm $A$ that answers the question “Does $M$ halt on $w$?”
  i.e. Suppose $A$ accepts the language $\{<M> : M \text{ halts on } w \}$
- Then take $<M>$, construct $M'$ from $M$ as before and create a Chapter 8 encoding $M''$ of $M'$ to feed to $A$
getting to the language question

- \( A \) accepts \(<M''>\) if and only if \( w \in L(M) = L(M') = L(M'') \) since \( M'' \) halts on \( w \) if and only if \( M'' \) accepts \( w \)
- Thus we get an algorithm for \{\(<M> : w \in L(M)\}\) which is impossible

another result related to emptiness

- Similarly, if we had an algorithm for “Does \( M \) halt on some input?”
- we would deduce an algorithm for \( L_{ne} = \{<M> : L(M) \text{ is not empty}\} \) which does not exist

halting on some string

- Suppose we had an algorithm \( B \) that answers the question “Does \( M \) halt on some \( w \)?”
  i.e. Suppose \( B \) accepts the language \{\(<M> : M \text{ halts on some input}\}\)
- Then take any \(<M>\), construct \( M' \) from \( M \) as before and create a Chapter 8 encoding \( M'' \) of \( M' \) to feed to \( B \)

Conversion to a known non-trivial problem

- \( B \) accepts \(<M''>\) if and only if \( L(M) = L(M') = L(M'') \) is not empty, since \( M'' \) halts on a string if and only if \( M'' \) accepts that string
- Thus we get an algorithm for \{\(<M> : L(M) \text{ is not empty}\}\) which is impossible
Rice's theorem for recursive index sets

- It is more complicated to find out whether $L_S$ is or is not rec. enum.
- Rice has a theorem: $L_S$ is rec. enum. if and only if $S$ satisfies 3 conditions: