Next Tuesday, quiz definition

Th. There exists a r.e. set that is not decidable.

Auxiliary result: There is a computable 1-1 correspondence between natural numbers and pairs of natural numbers

\[(0,0) \rightarrow (0,1) \quad (0,2) \rightarrow (0,3) \quad (0,4) \rightarrow (0,5) \]
\[(1,0) \rightarrow (1,1) \quad (1,2) \rightarrow (1,3) \quad (1,4) \rightarrow (1,5) \]
\[(2,0) \rightarrow (2,1) \quad (2,2) \rightarrow (2,3) \quad (2,4) \rightarrow (2,5) \]
\[(3,0) \rightarrow (3,1) \quad (3,2) \rightarrow (3,3) \quad (3,4) \rightarrow (3,5) \]

\[n \leftrightarrow (a_n, b_n)\]

\[S = \{ n : a_n \text{ is a code of a Java Program and } \text{it halts on } b_n \}\]

We need to prove 2 things:

- \(S\) is not decidable
- \(S\) is r.e.

Lemma: \(S\) is not decidable

By contradiction. Assume \(S\) is decidable. This means that there is an algorithm that, given a natural number \(n\), checks if \(n \in S\) or not.

Building a halting checker:

\[(p, d) \rightarrow (p, d) = (a_n, b_n)\]

we check whether \(n \in S\)

whether \(a_n \equiv p\)

\(\text{halts on } b_n \equiv d\)

Because the halting th. says we can’t build an algorithm that given \(p\) and \(d\) decide it.

\(p \text{ halts on } d, \ S\) is not decidable

Let’s prove that \(S\) is r.e.

Take pairs \((p, d)\) for which \(a_n \equiv p\) and \(b_n \equiv d\). On such \((p, d)\) steps...
Take pairs \((p, d)\) for which
- \(p \leq 1\) and \(d \leq 1\) and run each for 1 hr
- if one of them halts, we print \(n\) wrt. \(z\) \((p, d)\)

Take pairs \((p, d)\) for which \(p > 2\), \(d > 2\)
- and run each for 2 hr if some of them halts, we print a cor. to these halting pairs

\[
\begin{align*}
\text{pr.} \emptyset & \text{ on data } \emptyset \text{ for } 1 \text{ hr} & \text{1st stage} \\
\text{pr.} \emptyset & \text{ on data } 1 \\
\text{pr.} 1 & \text{ on data } 1 \\
\text{pr.} 1 & \text{ on data } \emptyset \\
\text{pr.} \emptyset & \text{ on data } \emptyset \text{ for } 2 \text{ hrs} & \text{2nd stage} \\
\text{pr.} 2 & \text{ on data } \emptyset \text{ for } 2 \text{ hrs} \\
\text{pr.} \emptyset & \text{ on data } 2 \text{ for } 2 \text{ hrs}.
\end{align*}
\]

Th.: There exists a r.e. set which is not decidable (called \(H\))

Q: Can we find a r.e. set whose complement is not r.e.?
   Yes, if \(H\) was r.e. then \(\overline{H}\) would be decidable, but we know that \(H\) is not decidable. So \(\overline{H}\) is not r.e.

\[\mathcal{L} = \{ \langle n, a_n \rangle : \text{a code of a Java program and an halts on } b_n \}\]

- \(A \cup B\) are decidable: \(A\overline{B}\) are r.e.
- \(A \cup B\) is decidable: \(A\overline{B}\) is r.e.
- \(A \cap B\) is decidable: \(A\overline{B}\) is r.e.
- \(A\overline{B}\) is decidable: \(A\overline{B}\) is r.e.
- \(A\cap B\) is decidable: \(A\overline{B}\) is r.e.
- \(A\cup B\) is decidable: \(A\overline{B}\) is r.e.
- \(A\cup B\) is decidable: \(A\overline{B}\) is r.e.

\[\mathcal{L} = \{ \langle n, a_n \rangle : \text{a code of a Java program and an halts on } b_n \} \]

\(A\overline{B}\) are decidable: \(A\cap B\) are r.e.
\(A \cup B\) is r.e.

- \(A\cap B\) is decidable: \(A\overline{B}\) is r.e.
- \(A\cup B\) is decidable: \(A\overline{B}\) is r.e.

\(A\cup B\) is decidable: \(A\overline{B}\) is r.e.

\(A\cup B\) is decidable:

\[A \subseteq N \text{ and } B = N\]
\( A \cup B \) is not decidable
\[ A = \emptyset \text{ and } B = \mathbb{N} \]
empty set \( \rightarrow A \cup B = \mathbb{N} \)

\( A \cap B \) is it r.e.? Yes because if \( A \) is decidable
\( A \) is also r.e.

is it decidable? \( \checkmark \) Yes \( A = B = \mathbb{N} \)

No \( B = \mathbb{N}, A = \emptyset \)
\( A \cap B = \emptyset \)

Another negative result

Check that the program always produces a correct result?

Def. A zero checker is an algorithm that given a program \( p \) that always halts, checks whether \( p \) always produces zero.

\[
\text{zero-checker} = \begin{cases} 
    \text{Yes, if } & \forall n (p(n) = 0) \\
    \text{No, if } & \exists n (p(n) \neq 0) \\
    \text{whether } & \text{if } \exists \text{ sometimes doesn't halt}
\end{cases}
\]

Proof Let’s assume that zero-checker exists.

Build a halt-checker based on it

Brainstorming

\( p \) halts on \( d \) means that there exists a moment \( t \) at which \( p \) halts on \( d \).

\( (p, d, t) \rightarrow \) whether \( p \) halts
on d by moment t? => yes, we test it
run to see if it
halts or don't

\[ f_{p,d}(t) = \begin{cases} \text{true} & \text{if } p \text{ halts on } d \text{ by moment } t \\ \text{false} & \text{otherwise} \end{cases} \]

\[ f_{p,d} \] is computable, halts always. We can apply a zero-checker to \( f_{p,d} \):

\[ \text{zero-checker}(f_{p,d}) = \begin{cases} \text{true} & \text{if } \forall t (f_{p,d}(t) = \emptyset) \iff p \text{ did not halt on } d \\ \text{false} & \text{if } \exists t (f_{p,d}(t) \neq \emptyset) \iff p \text{ halted on } d \end{cases} \]

\[ \text{halt-checker}(p,d) = \neg \text{zero-checker}(f_{p,d}) \]

\[ \begin{cases} \text{true} & \text{if } p \text{ halts on } d \\ \text{false} & \text{if } p \text{ doesn't on } d \end{cases} \]

zero-checker does not exist because
halt-checker doesn't exist.
- We assumed that zero-checker is possible.
- We concluded that a halt-checker was possible.
- We have proven that halt-checkers are not possible.
- This means our assumption is false.
- Thus, zero-checkers are not possible.

**Homework**

Give an example of \( A \in \text{RE} \) such that \( A \) is decidable \( B \in \text{RE} \) is r.e. \( C \in \text{R} \) and \( A \cup B \cup C \) is not decidable.

\( A \cap B \cap C \) is not decidable.

This is not halting-checker possible.
Theorem. No factorial checker is possible

\[
\text{fact-checker}(p) = \begin{cases} 
\text{true} & \text{if } \forall n (p(n) = n!) \\
\text{false} & \text{if } \exists n (p(n) \neq n!) 
\end{cases}
\]

Reduce to zero-checker

\[ q \quad ? \quad \forall n (q(n) = 0) \]
\[ p \quad \forall n (p(n) = n!) \]

\[ p(n) = q(n) + n! \]

He will explain again on Tuesday.