

Theorem: We can define a language D that we cannot decide (or even recognize).

Proof: Define $D = \{ x \in \{0,1\}^* : x \text{ compiles to a TM } M_x \text{ that does not accept } x \}$.
eg. using the Turing Kit, with x on ASCII file

Suppose D were recognizable. Then there would be a TM Q such that $L(Q) = D$.
(Q for "Quixote": It does not really exist)

Then Q would have a valid code $q \in \{0,1\}^*$ (in the "Turing Kit", indeed many such codes).
Don Quixote

Then $q \in D \iff Q \text{ accepts } q \text{ by defn of } L(Q) = D$
 $\iff Q \text{ does not accept } q \text{ by defn of } D, \text{ since } q \text{ compiles to } Q.$

This is a contradiction, because a statement may never be equivalent to its negation.
 Thus we must roll back to D not being recognizable. Hence D is not decidable either. \square

Compare: Turing
 ① num
 Ident
 D be
 Eg. Be
 ② The
 We can f
 Proof: Def
 If D were in P

Compare: Turing's form: The code of D as an infinite "binary decimal"

① number is mathematically definable but not computable.
 Identify binary strings x with natural numbers $1, 2, 3, \dots$
 $\epsilon 0100$
 D becomes the number d whose x th place is 1 if $x \in D$, 0 otherwise

Eg. PRIMES = $0.01101010001\dots$
2 3 5 7 11 ... n 2^A

② Theorem: For any function f from a set A to its power set $\mathcal{P}(A)$
 We can find a set $D \subseteq A$ that is not in the range of f . $\{B : B \subseteq A\}$

Proof: Define $D = \{ a \in A : a \text{ is not in the set } f(a) \}$.
 If D were in $\text{Range}(f)$, there would be an $a \in A$ st. $f(a) = D$. But then $a \in D \iff$

en recognize).
 is not accept x }.
 D. (Q for "Quixote": It does not really exist)
 ks). Don Quixote
 ites to Q .
 an
 de either. \square

Missing text at bottom right says "you finish the proof" (as a self-study exercise).

Theorem: A_{TM} is (recognizable but) not decidable.

Proof: Suppose we had a TM $M_0' \in \mathcal{U}(M_0) = A_{TM}$ and $\forall \langle M \rangle$ and w , $M_0'(\langle M \rangle w)$ always halts.

Then we would get a decider M_0' for \bar{D} that on any input x would run $M_0'(x, x)$.

But then the TM M_0 obtained by interchanging q_{acc} and q_{rej} in M_0' would be a decider for D .

Contradicting

Suppose D were recognizable. Then there would be a TM Q such that $L(Q) = D$.

Then Q would have a valid code $q \in \{0,1\}^*$ (in the "Turing kit" indeed many such codes).

Then $q \in D \iff Q$ accepts q by defn of $L(Q) = D$
 $\iff Q$ does not accept q by defn of D , since q computes to Q .

This is a contradiction, because a statement may never be equivalent to its negation.
 Thus we must roll back to D not being recognizable. Hence D is not decidable either. \square

Hence, the complement language $A_{TM} = \bar{D}$

Language of the Acceptance Problem For TMs

$\bar{D} = \{ \dots \}$

Further convention:
 $\langle M \rangle$ stands for some string x such that x computes to M and $\langle M \rangle w$ can be viewed as M .

Theorem: A_{TM} is (recognizable but) not decidable.

Proof: Suppose we had a TM $M_0' \in \mathcal{U}(M_0) = A_{TM}$ and $\forall \langle M \rangle$ and w , $M_0'(\langle M \rangle w)$ always halts.

Then we would get a decider M_0' for \bar{D} that on any input x would run $M_0'(x, x)$.

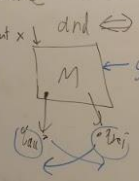
But then the TM M_0 obtained by interchanging q_{acc} and q_{rej} in M_0' would be a decider for D .

Contradicting

Suppose D were recognizable. Then

Theorem: A language L is decidable \iff its complement \bar{L} is decidable
 and $\iff L$ and \bar{L} are both recognizable.

Proof: Given a decider M for L , the machine M' obtained by switching q_{acc} and q_{rej} decides \bar{L} because M is total.



Proof of \Leftarrow : Monday

Hence, the complement language $A_{TM} = \bar{D}$

Language of the Acceptance Problem For TMs

$\bar{D} = \{ x : x \# x \}$

Further convention:
 $\langle M \rangle$ stands for some string x such that x computes to M and $\langle M \rangle w$ can be viewed as M and w .
 Here # is a string that $x \# x$
 $0 \rightarrow 0$
 $1 \rightarrow 1$
 $w \rightarrow 0$