

Decision Problems - (Poly-time) Decidable and Undecidable.

We can state ATM as a Decision Problem:

INSTANCE: A DTM M and an input string $w \in \Sigma^*$ to M .

QUESTION: Does M accept w ?

The language L is the set of (codes of) instances for which the answer is yes.

$L_{\text{ATM}} = \{ \langle M, w \rangle : M \text{ accepts } w \}$. Could be called " L_{ATM} " but just say L_{ATM} for the language too.

Naming convention: Type of problem subscripted by type of machine or formal object it applies to.

Algorithm: $L_{\text{ATM}} = \{ \langle M, w \rangle : M(w) \text{ accepts } w \}$ "Just run $M(w)$ ".

Algorithm: $L_{\text{DFA}} = \{ \langle M, w \rangle : M \text{ accepts } w \}$ \rightarrow (1) Convert N to an equivalent DFA M . Then run $M(w)$. This due $A_{\text{DFA}} \leq_m A_{\text{ATM}}$.

Algorithm: $L_{\text{NFA}} = \{ \langle N, w \rangle : N \text{ accepts } w \}$ More efficient: (2) Simulate $N(w)$ directly via $f(\langle N \rangle)$ = $\{ w_i \in \Sigma^* : N(w_i) \text{ accepts } w_i \}$ technically $f(\langle N \rangle)$ in the NFA-to-DFA proof.

$O(|w| \cdot |Q|)$ time.

Problems

"NE"

NE_DFA: INST: Just a DFA M (no string w)
QUES: Is $L(M) \neq \emptyset$?

NE_NFA: INST: An NFA N
QUES: Is $L(N) \neq \emptyset$.

DTM QUES: Is $L(m) \neq \emptyset$?

We will show this undecidable by showing $A_{\text{ATM}} \leq_m \text{NE}_{\text{DTM}}$ and using the contrapositive of Monday's theorem:

Suppose $A \leq_m B$. Then:

- if A is undecidable, B is undecidable.
- if A is not c.e., then B is not c.e.
- if A is not co-c.e., then B is not co-c.e.

A decision procedure for NE_{NFA} uses the fact that $L(N) \neq \emptyset \iff$ there is a path in the state graph of N from s to some accepting state $g \in F$. (we don't care about the chars or ϵ 's on the path)

Algorithm: Use Breadth-first Search and accept $\langle N \rangle$ if when a state $g \in F$ is found. This due $A_{\text{NFA}} \leq_m A_{\text{DFA}}$ since a DFA is a NFA.

NE_DFA follows by restriction since a DFA is a NFA.

The missing text for NE_{DFA} at top is "INST: Just a DFA M (no string w)" and for NE_{DTM} the missing text is "INST: A deterministic TM M ".

"Instance Tape" of the Source Problem A_{TM} is $\langle M, w \rangle$

Instance Tape of the Target Problem NE_{TM} (Construction) is "Just a Machine" M'

$\langle M, w \rangle \xrightarrow{f} M'$

A machine on a string. Logical Form: M'
must obey the requirement of the reduction.

$\langle M, w \rangle \in A_{TM} \iff \langle M' \rangle \in NE_{TM}$

"Unpack" the meanings $\begin{array}{ccc} M & \xrightarrow{\quad} & M' \\ \text{accepts } w & \xrightarrow{\quad} & L(M') \neq \emptyset \end{array}$

\therefore We need to show how (given any M, w) to build M' such that $L(M') \neq \emptyset$ if M accepts w .

AOE to diagram M' as a flowchart.

M and w can be part of the code of M' , which we could call $M'_{(M,w)}$

↓ input x
(Ignore x for the time being)

Write w after x
Simulate $M(w)$ open-endedly.
if and when M accepts w

Accept x .

This is a computable code mapping, so $f(M, w)$ is a computable function.

Computability flexibility

② correctness:
 • If M accepts w , then M accepts every string, so $L(M) = \Sigma^* \neq \emptyset$
 • But if M does not accept w , M' never accepts x , and since x is separate, this means $L(M') = \emptyset$.

$\therefore \langle M, w \rangle \in A_{TM} \iff f(M, w) \in NE_{TM}$

Therefore the NE_{TM} problem is undecidable.

But, the NE_{TM} language is c.e.: We can build an NTM N s.t. $L(N) = L_{NE_{TM}}$. ↓ input M

Theorem: For every NTM N we can build a DTM M s.t. $L(M) = L(N)$, by using Turing Kit to try all branches of N and converting that Java code to a DTM.

But, the NE_{TM} language N s.t. $L(N) = L_{NE_{TM}}$

Theorem: For every NTM N we can build a DTM M s.t. $L(M) = L(N)$, by using Turing Kit to try all branches of N and converting that Java code to a DTM.

ng) \therefore If M accepts w , then M' accepts every string, so $L(M) = \Sigma^* \neq \emptyset$ because when $L(M) = \Sigma^*$, we will show this undecidable by "nilly-nilly it accepts its own code (M') ".

But if M does not accept w , then M' never accepts x , and since x is separate, this means $L(M') = \emptyset$.

$\therefore \langle M, w \rangle \in A_{TM} \iff f(M, w) \in NE_{TM}$. Therefore the NE_{TM} problem is undecidable.

But, the NE_{TM} language is c.e.: We can build an NTM N s.t. $L(N) = L_{NE_{TM}}$. ↓ input M

Theorem: For every NTM N we can build a DTM M s.t. $L(M) = L(N)$, by using Turing Kit to try all branches of N and converting that Java code to a DTM.

Suppose $A \leq_m B$. Then:

- If A is undecidable, B is undecidable.
- If A is not c.e., then B is not c.e.
- If A is not 10-L.e., then B is not 11-L.e.

Diagram illustrating the many-to-one reduction $A \leq_m B$:

Input x is guessed and run in M . If it accepts, accept $\langle M \rangle$.

Diagram illustrating the many-to-one reduction $A \leq_m B$:

Input x is guessed and run in M . If it accepts, accept $\langle M \rangle$.