CS605
12 March 2025

HALTS = $\{\langle M, w\rangle$: M is a TM that halts on input word $w\}$

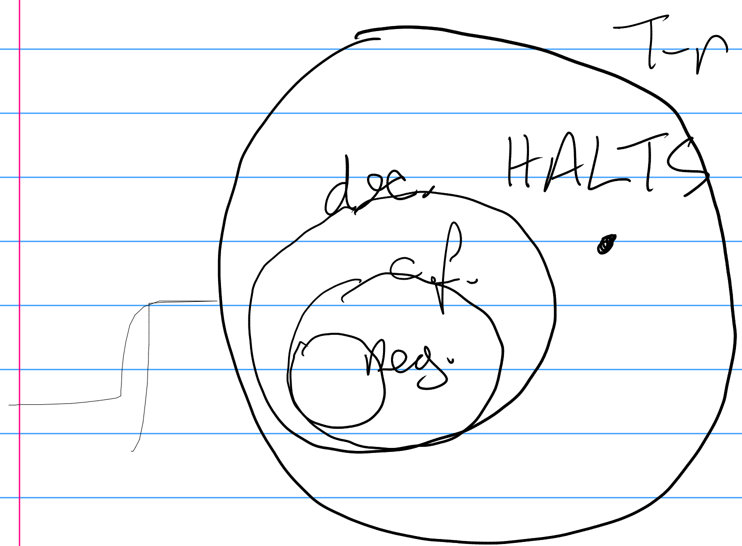
Prove that HALTS is T-r.

Proof We will prove this by constructing a TM N to recognise HALTS.

N = " On input $\langle M, w\rangle$:

1. Run M on w.
2. Accept."

TM N recognises the language therefore this proves that HALTS is T-r.



REJ = $\{\langle M, w \rangle$: M is a TM that rejects input word w$\}$

Prove that REJ is T-r.

~~Proof~~ We will prove this by constructing a TM N to recognise REJ.

N = " On input $\langle M, w \rangle$:

1. Run M on w.
2. If M rejects, accept."

TM N recognises the language therefore this proves that REJ is T-r.

3.2

VARIANTS OF TURING MACHINES

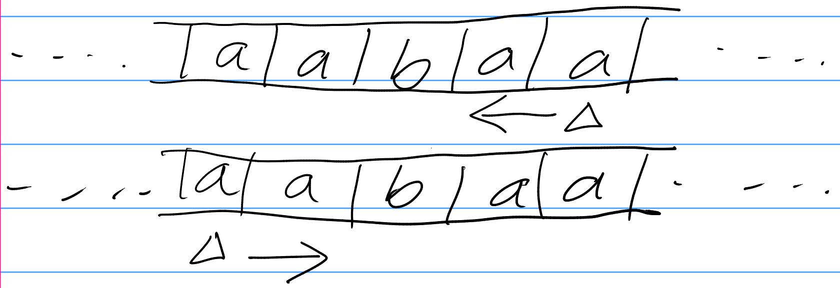
THEOREM 3.13 ....

Every multitape Turing machine has an equivalent single-tape Turing machine.

$$L = \{ w : w \in \{a,b\}^*, \ w = w^R \}$$

aababaa

Single Tape back & forth.



Two tapes, just 2 passes

Q	R	Q'	W	M
00	$\langle a,a \rangle$	00	$\langle a,a \rangle$	$\langle R,S \rangle$
00	$\langle a,b \rangle$	00	$\langle a,b \rangle$	$\langle R,S \rangle$
00	$\langle b,a \rangle$	00	$\langle b,a \rangle$	$\langle R,S \rangle$
00	$\langle b,b \rangle$	00	$\langle b,b \rangle$	$\langle R,S \rangle$
00	$\langle \_,a \rangle$	01	$\langle \_,a \rangle$	$\langle L,S \rangle$
00	$\langle \_,b \rangle$	01	$\langle \_,b \rangle$	$\langle L,S \rangle$
01	$\langle a,a \rangle$	01	$\langle a,a \rangle$	$\langle L,R \rangle$
01	$\langle b,b \rangle$	01	$\langle b,b \rangle$	$\langle L,R \rangle$
01	$\langle \_,\_ \rangle$	99	$\langle \_,\_ \rangle$	$\langle L,R \rangle$

Loop first tape head to last symbol

$\delta : Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$

NONDETERMINISTIC TURING MACHINES

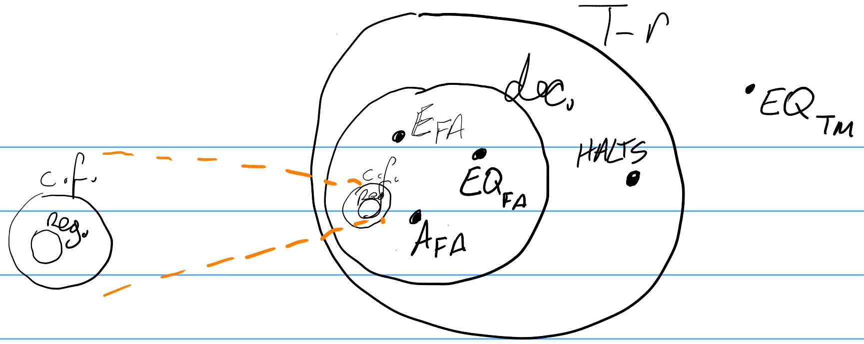
A nondeterministic Turing machine is defined in the expected way. At any point in a computation the machine may proceed according to several possibilities. The transition function for a nondeterministic Turing machine has the form

$$\delta \colon Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}).$$

The computation of a nondeterministic Turing machine is a tree whose branches correspond to different possibilities for the machine. If some branch of the computation leads to the accept state, the machine accepts its input. If you feel the need to review nondeterminism, turn to Section 1.2 (page 47). Now we show that nondeterminism does not affect the power of the Turing machine model.

THEOREM 3.16

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.



4.1

DECIDABLE LANGUAGES

In this section we give some examples of languages that are decidable by algorithms. We focus on languages concerning automata and grammars. For example, we present an algorithm that tests whether a string is a member of a context-free language (CFL). These languages are interesting for several reasons. First, certain problems of this kind are related to applications. This problem of testing whether a CFL generates a string is related to the problem of recognizing and compiling programs in a programming language. Second, certain other problems concerning automata and grammars are not decidable by algorithms. Starting with examples where decidability is possible helps you to appreciate the undecidable examples.

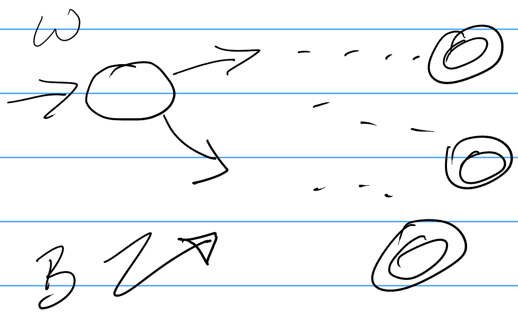
DECIDABLE PROBLEMS CONCERNING REGULAR LANGUAGES

$$A_{\mathsf{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts input string } w\}.$$

THEOREM 4.1 ....

A_{DFA} is a decidable language.

Proof idea

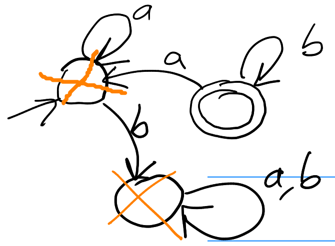


Proof: We will prove this by constructing a TM M to decide A_{FA}.

M = "On input $\langle B, w \rangle$, where B is a DFA and w is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state, *accept*. If it ends in a nonaccepting state, *reject*."

TM M decides A_{FA}, therefore this proves A_{FA} is decidable.



$$E_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}.$$

THEOREM 4.4

E_{DFA} is a decidable language.

Proof We will prove this by constructing a TM T to decide E_{FA}.

$T =$ "On input $\langle A\rangle$ where A is a DFA:

1. Mark the start state of A.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, *accept*; otherwise, *reject*."

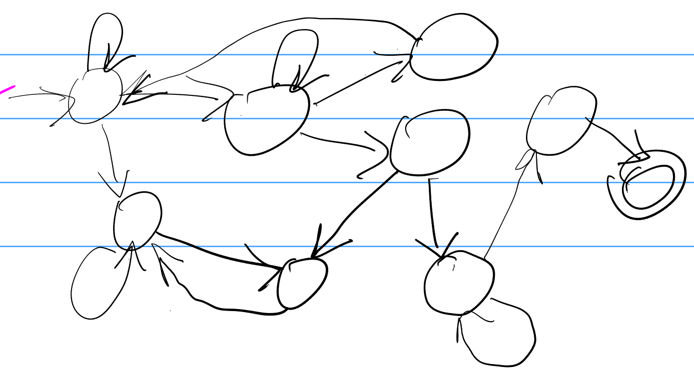
Very important — like { and } in Java.

Since TM T decides E_{FA}, this proves E_{FA} is decidable.

Example for yourselves:

$LT5 = \{\langle C\rangle :$ C is FA that accepts at least one word w where $|w| < 5\}$

Option 1



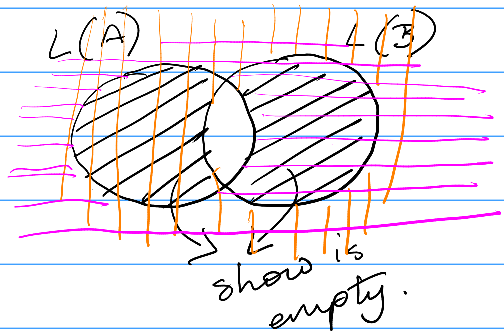
Option 2:
For w in Σ^4:
Run C on w

$$EQ_{\mathsf{DFA}} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}.$$

THEOREM 4.5

EQ_{DFA} is a decidable language.

Proof idea



where A and B are two FA.

Thm Regular languages are closed under union, intersection, and complement. [Not part of CS605]

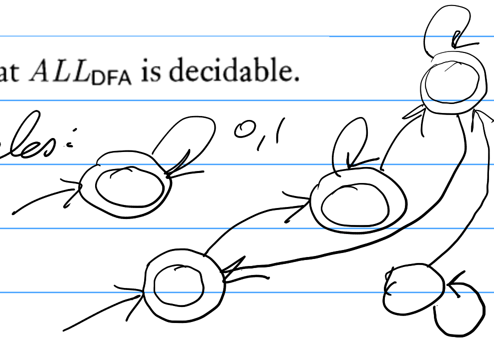
$$\left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

Not part of CS605

Example 4.3 in Sipser's book

4.3 Let $ALL_{\mathsf{DFA}} = \{\langle A\rangle \mid A$ is a DFA and $L(A) = \Sigma^*\}$. Show that ALL_{DFA} is decidable.

Examples:



Proof

We will prove this by constructing a TM M to decide ALL_{FA}.

M = "On input $\langle N\rangle$:

1. Mark the start state of N.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If any non accept state marked, reject, else accept."

Alternative:

4. If only accept states marked, accept, else reject."

M decides ALL_{FA}, therefore ALL_{FA} is decidable.

FYI: this would be incorrect as the middle part of the proof.

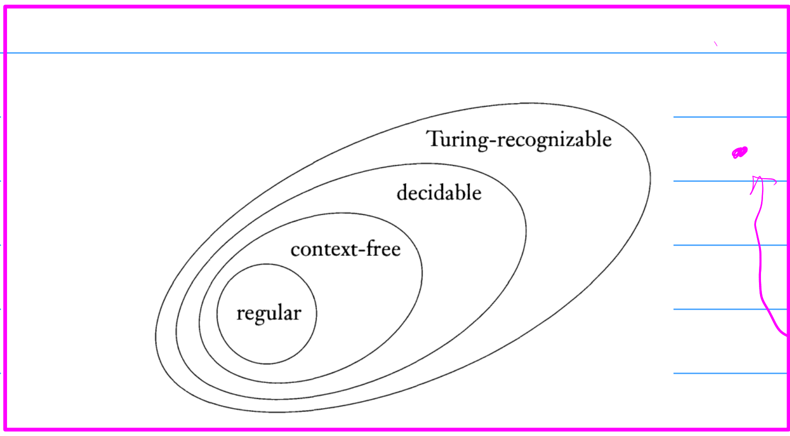
M = "On input $\langle N\rangle$:

1. For each word w in Σ^*:
2. Run N on w.
   If w is not accepted, reject
   else accept."

3. . . . . . .

This attempt would yield few marks because it is a recogniser, not a decider. Wherever we put the "else accept" code it will either be executed in the first iteration of the loop (as it is now) or never executed (if it was on line 3). This code is actually a pretty good recogniser for the complement of ALL_FA but it is not appropriate as a decider for ALL_FA.

2^{Σ^*}



Prove at least one language exists here, using diagonalisation.

FIGURE 4.10
The relationship among classes of languages

We will prove this using diagonalisation. Assume the set of all languages is countable. Then it possible to list them all in some order, representing each one by an infinite binary string denoting whether each word over the binary alphabet is in that language or not. This is shown below. Then we extract the diagonal.

Let $\Sigma = \{0, 1\}$

2^{Σ^*} \ Σ^*	e	0	1	00	01	10	11	000	
$\emptyset$	F	F	F	F	F	F	F	F	....
Σ^*	T	T	T	T	T	T	T	T	....
$0^n 1^n$	T	F	F	F	T	F	F	F	....
ww^R	T	F	F	T	F	F	T	F	....
$w = w^R$	T	T	T	T	F	F	T	T	....
⋮									

The diagonal language, and the language formed by changing each T to F and F to T.

F T F T F .....
T F T F T ...... $\Rightarrow$ Not in list A contradiction.

Such a language not in the list can be found no matter how we try to order the set of all languages. Therefore the set of all languages is uncountable.

However, the set of all Turing machines languages is countable, and consequently the set of all languages recognised by Turing machines (the set of Turing-recognisable languages) is also countable.
Therefore, there must exist some languages (actually, most of them) that are not Turing-recognisable.