To prove X is decidable/recognizable:

1. X ≤ Y where Y is decidable/recognizable.
2. Turing Machine that decides/recognizes.

To prove X is not decidable/recognizable:

1. Proof by contradiction.
2. Y ≤ X where Y is not decidable/recognizable.

Terms:

1. ≤ = mapping reducible.
2. Σ = Input alphabet.
3. ⊆ = subset of
4. ⟨asdf⟩ = Input-ify whatever is inside angle brackets.
5. Computable Function = function that takes an input, gives an output, always halts.
6. Mapping Reducible (≤) = Takes any member of A and translates it to some member of B.

Steps To Prove Undecidable/Non-Turing Recognizable:

1. Choose a known undecidable problem.
2. Describe an instance of the known undecidable problem.
3. Describe an instance of the subject problem.
4. Provide a computable mapping function from an instance of the known undecidable problem to an instance of the subject problem.

4a. Argue that this function is computable.

5. Prove that *w* ∈ A => f(*w*) ∈ B.
6. Prove that *w* !∈ A => f(*w*) !∈ B or f(*w*) ∈ B => w ∈ A.
7. Conclusion.

Known Undecidable Problems:

1. HALT = {⟨*M, w*⟩| *M* is a TM and *M* halts on input *w*}.
2. A = {⟨*M, w*⟩| *M* is a TM and *M* accepts *w*}. NOTE: A does the opposite of its inner input. (contradiction).
3. E = ⟨*M*⟩ Where M is a Turing Machine with the language: L(M) = ø.
4. Regular
5. EQ
6. A complement = {⟨*M, w*⟩| *M* is a TM and *M* rejects *w*}

Theorems

1. Theorem 5.22: If A ≤ B and B is decidable, then A is decidable.
2. Corollary 5.23: if A ≤ B and A is undecidable, then B is undecidable.
3. Theorem 4.4 E_DFA is a decidable language that decides any real DFAs.
4. Theorem 1.54: Converts a RE to a NFA
5. Theorem 1.39: Converts a NFA to a DFA
6. Theorem 4.5: EQ_DFA; a Turing Machine that compares two DFA determining if they're equal.
7. Theorem 4.22: A language is decidable iff it's Turing-recognizable & Co-turing-recognizable. Therefore A_TM compliment is not turing recognizable.
8. Corollary 5.29: if A ≤ B and A isn't Turing-recognizable, then B isn't Turing-recognizable.
9. Theorem 4.1: A_DFA is a decidable language.

Examples from 5a.

Let T = {⟨M ⟩|M is a TM that accepts wR whenever it accepts w}. Show that T is undecidable.

Solution:

Proof Idea: use Corollary 5.23. We have a collection of known undecidable languages from the text book and assignments.. We need to choose any one of them as A, and provide a computable function (a halting Turing-machine that translates input to output) that will convert *any i*nstance of A into *some* instance of B, and *any* non-instance of A into *some* non-instance of B.

Proof:

Choose an undecidable problem, describe an instance:

From Theorem 4.11, ATM is undecidable. instance of ATM = ⟨M, w⟩, where M is a TM and w is a string accepted by M.

Describe an instance of the subject language:

An instance of T = ⟨M *′*⟩ where M *′* is Turing machine that accepts wR if it accepts w.

Describe the computable function

Our computable function must receive ⟨M, w⟩ as input and return ⟨M *′*⟩ as output. Such that the input ⟨M, w⟩∈ATM if and only if the output ⟨M *′*⟩∈T .

The M *′* we construct only needs to be some instance of T . We choose to create a machine that recognizes the language {w, wR} if M accepts w, and the language {wR} if M does not accept w.

Let F = "On input ⟨M, w⟩:

1. Construct the following machine M *′*.

    M *′* = "On input x:

    1. If x != w and x != wR, reject.
    2. If x = wR, accept.
    3. If x = w and M accepts w, accept ; otherwise reject."

2. Output ⟨M *′*⟩"

F 's work of constructing M ′ is finite, it will halt. F is a computable function.
If ⟨M, w⟩∈ATM, then ⟨M ′⟩∈T
If ⟨M, w⟩∈ATM, the L(M ′) = w, wR. This meets the requirements for membership in T .
If ⟨M, w⟩!∈ATM, then ⟨M ′⟩!∈T
If ⟨M, w⟩!∈ATM, the L(M ′) = wR. This does not meet the requirements for membership in T .
Conclusion
Thus ⟨M, w⟩∈ATM ⇔⟨M ′⟩∈T . As noted in the proof idea, Corollary 5.23 states that T is undecidable.

Example from 4a:

Consider the problem of determining whether a DFA and a regex are equivalent. Express the problem as a language and show that it is decidable.

Solution:
Let A = { ⟨D, R⟩| D is a DFA R is a regex L(D) = L(R)}
Let M_DFA/REX = on input ⟨D, R⟩where D is a DFA and R is a regex:

1. Construct an NFA N from R using theorem 1.54.
2. Construct DFA E from N using theorem 1.39.
3. Run TM F from theorem 4.5 on ⟨D, E⟩that decides EQ_DFA.
4. If F accepts, accept. If F rejects, reject.

The conversion procedure halt because they are finite. F halts, because it is a decider. M_DFA/REX halts and is a decider.

FINAL EXAM QUESTIONS:

How do you prove a language is not regular? - pumping lemma

What can and cannot be computed by turing machines?

An alphabet is a finite set of symbols: E

A string w is a sequence of symbols from E

The infinite set of all possible strings is E*

A string w inside E* has finite length

A language is a subset of possible strings L subset E*

A language may be finite or infinite however L union Lcomplement = E* and is finite.

A string is either in a language or is not in a language.

Computable problems can be defined in terms of determining whether or not a string is a member of a language w inside L

A computing machine defines a process for determining w inside L (turing machine)

A generator defines a process to generate all w inside L (NFA DFA RE CFG)

Languages are classified based on the computing machine structures needed to compute the membership of a string or the generator structures needed to generate the members of a language

Language classes

Not turing recognizable

Turing recognizable

Co-turing recognizable

Not co-turing recognizable

Decidable (turing-decidable)

Context-free

Regular