COMP481 Review Problems
Turing Machines and (Un)Decidability
Luay K. Nakhleh

NOTES:

1. In this handout, I regularly make use of two problems, namely

   - The Halting Problem, denoted by HP, and defined as
   $$HP = \{\langle M, w\rangle | M \text{ is a TM and it halts on string } w\}.$$
   - The complement of the Halting Problem, denoted by $\overline{HP}$, and defined as
   $$\overline{HP} = \{\langle M, w\rangle | M \text{ is a TM and it does not halt on string } w\}.$$

2. I use "decidable" and "recursive" interchangeably, and use "semi-decidable" and "recursively enumerable" interchangeably. I use R for recursive, and RE for recursively enumerable.

3. Recall: "$\leq_m$" denotes *mapping reducibility*. For example, $L_1 \leq_m L_2$ means "L_1 mapping reduces to L_2".

4. To show a language L is not in R (not in RE), it suffices to show that $L' \leq_m L$ for some L' that is not in R (not in RE). From the theorem we've learned, it follows that L is not in R (not in RE).

5. Showing $L' \leq_m L$ is equivalent to showing $\overline{L'} \leq_m \overline{L}$.

6. Rice's Theorem (the version I use):

   "Let C be a proper, nonempty subset of RE. Then, the language
   $$L_C = \{\langle M\rangle \ : \ L(M) \in C\}$$
   is not recursive."

   So, to show a language L is not recursive using Rice's Theorem, I pick the appropriate C (i.e., a C such that $L_C = L$), prove that (1) $C \subseteq RE$, (2) $C \neq RE$, and (3) $C \neq \emptyset$. Finally, I conclude that L_C is not recursive.

THE PROBLEMS:

1. For each of the following languages, state whether each language is (I) recursive, (II) recursively enumerable but not recursive, or (III) not recursively enumerable. Prove your answer.

   - $L_1 = \{\langle M\rangle | M \text{ is a TM and there exists an input on which } M \text{ halts in less than } |\langle M\rangle| \text{ steps}\}$.
     - **R.** M^* that decides the languages works as follows on input $\langle M\rangle$. It first finds the length of $\langle M\rangle$, and stores it. Then, it runs M on all inputs of length at most $|\langle M\rangle|$, for at most $|\langle M\rangle|$ steps, and accepts if M accepts at least one of the strings within the specified number of steps.

       You might wonder why we limited the length of the strings. Since we bound the number of steps that M runs on an input, then there is no point on looking at any strings that are longer

than that number, since if a TM is allowed to run for at most c steps, it is not possible for that TM to "process" any input symbol beyond the c^{th} symbol!
The number of possible inputs is finite, and the number of steps M runs on each input is finite, therefore M is guaranteed to halt and decide the language.

- $L_2 = \{\langle M\rangle | M$ is a TM and $|L(M)| \leq 3\}$.
  - **Not RE.** We prove this by a reduction from $\overline{HP}$. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w: it erases its input, copies M and x to its tape, and runs M on x; it accepts if M halts on x.
    We now prove the validity of reduction:
    * $\langle M, x\rangle \in \overline{HP} \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept any input $\Rightarrow |L(M')| \leq 3 \Rightarrow M' \in L_2$.
    * $\langle M, x\rangle \notin \overline{HP} \Rightarrow M$ halts on $x \Rightarrow M'$ accepts all inputs $\Rightarrow |L(M')| > 3 \Rightarrow M' \notin L_2$.
  - **GOOD EXERCISE:** What happens if the property was $|L(M)| = 2$? Prove your answer.
- $L_3 = \{\langle M\rangle | M$ is a TM and $|L(M)| \geq 3\}$.
  - **RE.** M^* that semidecides the language, runs M on all inputs in an interleaved mode, and halts whenever 3 inputs have been accepted. Notice that M^* generates the input strings for M one by one as they are needed (It is not allowed that M^* first generates all strings, and then starts running M on them, since generating the inputs takes infinite time!).
  - **Not R.** We prove this by a reduction from HP. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w works as follows: It erases w, puts M and x on its tape, and runs M on x and accepts if M halts on x.
    We now prove the validity of the reduction:
    * $\langle M, x\rangle \in HP \Rightarrow M$ halts on $x \Rightarrow M'$ accepts all inputs $\Rightarrow |L(M')| \geq 3 \Rightarrow M' \in L_3$.
    * $\langle M, x\rangle \notin HP \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept any input $\Rightarrow |L(M')| < 3 \Rightarrow M' \notin L_3$.

not recursively enumerable
we will reduce ~HP<={M| M is a
TM that accepts all even numbers}

M' simulates M on x for |y| steps.
for any even number, |y|>=1
we are only considering the M's
that dont begin in an accept/reject
state since the halting problem
is decidable for them. Thus, even is
M halts on x, atleast one move is
required.
We reject y is M halts on x, else
accept y.
consider this=>M halts on x=> takes
>= 1 steps => rejects 0
=>M doesnt halt on x =>
accepts all inputs => accepts all even nuimbers

- $L_4 = \{\langle M\rangle | M$ is a TM that accepts all even numbers$\}$.
  - **Not RE.** We prove this by a reduction from $\overline{HP}$. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w: it runs M on x for $|w|$ steps; it rejects if M halts on x within $|w|$ steps, and accepts otherwise.
    We now prove the validity of the reduction:
    * $\langle M, x\rangle \in \overline{HP} \Rightarrow M$ does not halt on $x \Rightarrow M'$ accepts all inputs, and in particular, all even numbers $\Rightarrow M' \in L_4$.
    * $\langle M, x\rangle \notin \overline{HP} \Rightarrow M$ halts on x within k steps $\Rightarrow M'$ rejects all inputs w whose length is greater than or equal to $k \Rightarrow M'$ rejects an infinite number of even numbers $\Rightarrow M' \notin L_4$.
  - **GOOD EXERCISE:** What happens if the property was "M does not accept all even numbers"? What happens if the property was "M rejects all even numbers"? Prove your answers.
- $L_5 = \{\langle M\rangle | M$ is a TM and $L(M)$ is finite$\}$. not recursively enumerable, proof given in Kozen
  - **Not RE.** We prove this by a reduction from $\overline{HP}$. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w: it runs M on x.
    We now prove the validity of the reduction:
    * $\langle M, x\rangle \in \overline{HP} \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept any strings $\Rightarrow L(M')$ is finite $\Rightarrow M' \in L_5$.

very easy reduction from ~HP <= {M|L(M) is infinite}. consider M' that runs M on x for |y| steps and rejects in case M halts. then when M halts on input x=> only those strings get accepted where |y| < steps of computation(finite). so L(M') is infinite. in case m doesnt halt on x, everything gets accepted

  * $\langle M, x\rangle \notin \overline{HP} \Rightarrow M$ halts on $x \Rightarrow M'$ accepts all strings $\Rightarrow L(M')$ is infinite $\Rightarrow M' \notin L_5$.

- $L_6 = \{\langle M\rangle | M$ is a TM and $L(M)$ is infinite$\}$.
  - **Not RE.** We prove this by a reduction from $\overline{HP}$. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w: it runs M on x for $|w|$ steps; it rejects if M halts on x within $|w|$ steps, and accepts otherwise. We now prove the validity of the reduction:
    * $\langle M, x\rangle \in \overline{HP} \Rightarrow M$ does not halt on $x \Rightarrow M'$ accepts all strings $\Rightarrow L(M')$ is infinite $\Rightarrow M' \in L_6$.
    * $\langle M, x\rangle \notin \overline{HP} \Rightarrow M$ halts on x within k steps $\Rightarrow M'$ rejects all strings whose length is greater than or equal to $k \Rightarrow L(M')$ is finite $\Rightarrow M' \notin L_6$.
- $L_7 = \{\langle M\rangle | M$ is a TM and $L(M)$ is countable$\}$.
  - **R.** This is the language of all TM's, since there are no uncountable languages (over finite alphabets and finite-length strings).
- $L_8 = \{\langle M\rangle | M$ is a TM and $L(M)$ is uncountable$\}$.

recursive, since this simply rejects every turing machine

  - **R.** This is the empty set; there are no uncountable languages (over finite alphabets and finite-length strings).
- $L_9 = \{\langle M_1, M_2\rangle | M_1$ and M_2 are two TMs, and $\varepsilon \in L(M_1) \cup L(M_2)\}$.

we are assuming that the turing machines are not total.

  - **RE.** M^* that semidecides the language run the two machines on ε (it interleaves the run between the machines), and accepts if at least one of them accepts.
  - **Not R.** We prove this by a reduction from HP. $\tau(\langle M, x\rangle) = \langle M', M'\rangle$. M' on input w: it runs M on x and accepts if M halts on x.
    We now prove the validity of the reduction:
    * $\langle M, x\rangle \in HP \Rightarrow M$ halts on $x \Rightarrow M'$ accepts all strings, and in particular it accepts $\varepsilon \Rightarrow \varepsilon \in L(M') \cup L(M') \Rightarrow \langle M', M'\rangle \in L_9$.
    * $\langle M, x\rangle \notin HP \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept any strings, and in particular does not accept $\varepsilon \Rightarrow \varepsilon \notin L(M') \cup L(M') \Rightarrow \langle M', M'\rangle \notin L_9$.
- $L_{10} = \{\langle M_1, M_2\rangle | M_1$ and M_2 are two TMs, and $\varepsilon \in L(M_1) \cap L(M_2)\}$.

not recursive since again consider M1, M2 to be M', M'. then the intersection language is the language itself.
if the intersection language contains epsilon in this case, m halts. otherwise it doesnt.

re since we'll accept once both the machines have accepted.

  - **RE.** M^* that semidecides the language run the two machines on ε (it interleaves the run between the machines), and accepts if both of them accept.
  - **Not R.** We prove this by a reduction from HP. $\tau(\langle M, x\rangle) = \langle M', M'\rangle$. M' on input w: it runs M on x and accepts if M halts on x.
    We now prove the validity of the reduction:
    * $\langle M, x\rangle \in HP \Rightarrow M$ halts on $x \Rightarrow M'$ accepts all strings, and in particular it accepts $\varepsilon \Rightarrow \varepsilon \in L(M') \cap L(M') \Rightarrow \langle M', M'\rangle \in L_{10}$.
    * $\langle M, x\rangle \notin HP \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept any strings, and in particular does not accept $\varepsilon \Rightarrow \varepsilon \notin L(M') \cap L(M') \Rightarrow \langle M', M'\rangle \notin L_{10}$.

not recursively. consider M2 to be any automaton which doesnt accept epsilon, in that case L(M1) \ l(m2) is the same as l(m1).
so we are essentially asking if l(m1) contains epsilon which is not recursive.

- $L_{11} = \{\langle M_1, M_2\rangle | M_1$ and M_2 are two TMs, and $\varepsilon \in L(M_1) \setminus L(M_2)\}$.
  - **Not RE.** We prove this by a reduction from $\overline{HP}$. $\tau(\langle M, x\rangle) = \langle M_1, M_2\rangle$. M_1 is a TM that halts and accepts on any input (i.e., $L(M_1) = \Sigma^*$). M_2 on input w: it runs M on x and accepts if M halts on w.
    We now prove the validity of the reduction:

not RE
consider when m1 accepts epsilon trivially, we are essentially asking if m2 doesnt accept epsilon.
m2 runs m on x, and accepts if m halts on x.
m doesnt halt on x => m2 doesnt accept epsilon
reduction from ~hp.

- $\langle M, x\rangle \in \overline{HP} \Rightarrow M$ does not halt on $x \Rightarrow M_2$ does not accept any input $\Rightarrow L(M_1) \setminus L(M_2) = \Sigma^* \Rightarrow \varepsilon \in L(M_1) \setminus L(M_2) \Rightarrow \langle M_1, M_2\rangle \in L_{11}$.
- $\langle M, x\rangle \notin \overline{HP} \Rightarrow M$ halts on $x \Rightarrow M_2$ accepts all inputs $\Rightarrow L(M_1) \setminus L(M_2) = \emptyset \Rightarrow \varepsilon \notin L(M_1) \setminus L(M_2) \Rightarrow \langle M_1, M_2\rangle \notin L_{11}$.

- $L_{12} = \{\langle M\rangle | M$ is a TM, M_0 is a TM that halts on all inputs, and $M_0 \in L(M)\}$.
  - **RE.** M^* that semidecides the language runs M on M_0 and halts if M accepts M_0.
  - **Not R.** We prove this by Rice's Theorem. Let $C = \{L \in RE : M_0 \in L\}$.

    $C \subseteq RE$: trivial.

    $C \neq \emptyset$: e.g., $\Sigma^* \in C$.

    $C \neq RE$: e.g., $\emptyset \in RE \setminus C$.

    Therefore, $L_C = \{\langle M\rangle : L(M) \in C\} \notin R$. But, $L_C = \{\langle M\rangle : M_0 \in L(M)\}$; therefore, L_{12} is not recursive.

recursive

- $L_{13} = \{\langle M\rangle | M$ is a TM, M_0 is a TM that halts on all inputs, and $M \in L(M_0)\}$.
  - **R.** M^* that decides L_{13} runs M_0 on M and accepts if M_0 accepts M and rejects if M_0 rejects M. The difference between this and L_{12} is that here M_0, the machine that runs on the input, is guaranteed to always halt; however, in L_{12}, M, the machines that runs on the input, might not halt!

- $L_{14} = \{\langle M, x\rangle | M$ is a TM, x is a string, and there exists a TM, M', such that $x \notin L(M) \cap L(M')\}$.

  recursive since we can always make a turing machine that rejects x

  - **R.** For any TM, M, there is always a TM, M', such that $x \notin L(M) \cap L(M')\}$. In particular, take M' to be the machine that rejects all inputs. So, this is basically the language of all TM's.

recursive
try all the inputs of length <= 1000
accept if M halts on some input,
else reject

- $L_{15} = \{\langle M\rangle | M$ is a TM, and there exists an input on which M halts within 1000 steps$\}$.
  - **R.** The proof is very similar to L_1.

- $L_{16} = \{\langle M\rangle | M$ is a TM, and there exists an input whose length is less than 100, on which M halts$\}$.

  simultaneously try on different tracks all inputs of length<100
  accept if any of those halts.

  usual M'. ask if it halts on an input of <100. yes => M halts on x

  - **RE.** M^* that semidecides the language runs M on all strings of length at most 100 in an interleaved mode, and halts if M accepts at least one.
  - **Not R.** We prove this by a reduction from HP. $\tau(\langle M, x\rangle) = \langle M'\rangle$. M' on input w: it runs M on x.

    Prove the validity of this reduction! It's similar to the other reductions from HP that we used before.

should be RE since we can run the k turing machines simultaenously, and accept
as soon as k/2 of them halt.
not r.
suppose k/2 are set as m' and the rest are a trivial turing machine which loops on
all inputs => we are essentially asking if m' halts on x.
take the usual m', if the answer is yes => m halts on x

- $L_{17} = \{\langle M\rangle | M$ is a TM, and M is the only TM that accepts $L(M)\}$.

  recursive, since this is empty set, we can always make isomorphic turing machines

  - **R.** This is the empty set, since every language has an infinite number of TMs that accept it.

- $L_{18} = \{\langle k, x, M_1, M_2, \ldots, M_k\rangle | k$ is a natural number, x is a string, M_i is a TM for all $1 \leq i \leq k$, and at least $k/2$ TM's of $M_1, \ldots, M_k$ halt on $x\}$.
  - **RE.** M^* that semidecides the language runs the k machines (it interleaves the runs of the machines) on x, and accepts if at least $k/2$ of them accept.
  - **Not R.** We prove this by a reduction from HP. $\tau(\langle M, x\rangle) = \langle 2, x, M', M'\rangle$. M' runs M on x, and accepts if M halts on x. We now prove the validity of the reduction:
    - $\langle M, x\rangle \in HP \Rightarrow M$ halts on $x \Rightarrow M'$ accepts $x \Rightarrow \langle 2, x, M', M'\rangle \in L_{18}$.

- $\langle M, x\rangle \notin HP \Rightarrow M$ does not halt on $x \Rightarrow M'$ does not accept $x \Rightarrow \langle 2, x, M', M'\rangle \notin L_{18}$.

- $L_{19} = \{\langle M\rangle | M$ is a TM, and $|M| < 1000\}$.
  - **R.** In this question, we are talking about all the descriptions of Turing machines using a fixed alphabet (of finite size, of course), i.e., TM's that are encoded as input to the universal TM. So, L_{19} is finite, and hence recursive.

- $L_{20} = \{\langle M\rangle | \exists x, |x| \equiv_5 1$, and $x \in L(M)\}$.
  - **RE.** A TM, M^*, that semidecides the language runs M on all inputs (in an interleaved mode) x, such that $|x| \equiv_5 1$, and halts if M accepts at least one such string.
  - **NOT R.** Use Rice's theorem. Take $C = \{L \in RE | \exists x, |x| \equiv_5 1$, and $x \in L\}$. It's easy to prove that (1) $C \subseteq RE$, (2) $C \neq \emptyset$, and (3) $C \neq RE$. Hence, $L_C = \{\langle M\rangle | L(M) \in C\} = \{\langle M\rangle | \exists x, |x| \equiv_5 1$, and $x \in L(M)\} \notin R$.

not re
consider m' that operates
m on x for |y| steps and
accepts if m doesnt halt
now, if m doesnt halt on x=>
all strings, and hence all pal-
indromes are accepted
if m does halt =>
atleast some palindromes
will not be accepted, like
the palindrome with a single
input alphabet

- $L_{21} = \{\langle M\rangle | M$ is a TM, and M halts on all palindromes$\}$.
  - **NOT RE.** Some of you might have seen the application of Rice's theorem to this language, which proves the languages is not in R. Nevertheless, this languages is also not in RE, and we prove that using reductions. Remember: Rice's theorem does not prove a language is not in RE! We reduce $\overline{HP}$ to L_{21}. The reduction function τ is as follows: $\tau(\langle M, w\rangle) = \langle M^*\rangle$. M^* on input x works as follows: M^* runs M on w for $|x|$ steps: if M halts on w within $|x|$ steps, reject, otherwise accept.
    Prove the validity of the reduction.

a , a^2, a^4, ...,
consider m' that runs m on x
and accepts if m halts
m halts => no
m doesnt halt => yes
~hp <= l22

atleast k strings of the form a*b*
accepted by the machine
intuitively its not RE

- $L_{22} = \{\langle M\rangle | M$ is a TM, and $L(M) \cap \{a^{2^n} | n \geq 0\}$ is empty$\}$.
  - **NOT RE.** This language contains all TM's that do not accept any string of the form a^{2^n}. The proof is by a reduction from $\overline{HP}$. $\tau(\langle M, w\rangle) = M^*$.
    M^* on input x: erase x, and run M on w. If M halts on w then M^* accepts.
    Prove the validity of the reduction.

- $L_{23} = \{\langle M, k\rangle | M$ is a TM, and $|\{w \in L(M) : w \in a^*b^*\}| \geq k\}$.
  - **RE.** Easy; prove it yourself.
  - **Not R.** Also easy; prove it yourself.

- $L_{24} = \{\langle M\rangle | M$ is a TM that halts on all inputs and $L(M) = L'$ for some undecidable language $L'\}$.
  - **R.** Since M halts on all inputs, then $L(M)$ is decidable, and hence it can't be that $L(M)$ equals some undecidable language L'. Therefore, $L_{24} = \emptyset$ which is recursive.

- $L_{25} = \{\langle M\rangle | M$ is a TM, and M accepts (at least) two strings of different lengths$\}$.
  - **RE.** M^* that semidecided the language runs M on all inputs in an interleaved mode, halts and accepts once M accepts two strings of different lengths.
  - **NOT R.** Reduce the halting problem, or use Rice's theorem. Both are easy to use with this language.
  - **GOOD EXERCISE:** What happens if we replace "at least" by "exactly". Prove your answer.

not R
consider m' that
runs m on x, and if
m halts, accepts
string y of even length
both l(m) and ~l(m)
are infinite

intuitively not re
m' runs m on x for
|y| steps and accepts
if m doesnt halt and |y|
was even

m halts on x => l(m) is finite
~l(m) is infinite
m doesnt halt on x =>
l(m) is infinite, as is ~l(m)
~hp <= l26

- $L_{26} = \{\langle M\rangle | M$ is a TM such that both $L(M)$ and $\overline{L(M)}$ are infinite$\}$.

- **NOT RE.** Reduce $\overline{HP}$. $\tau(\langle M, w\rangle) = \langle M^*\rangle$.
  M^* on input x works as follows:
  if x is of odd length, accept.
  if x is of even length, run M on w. If M halts, accept.

  Let's prove the validity of this one.
  $\langle M, w\rangle \in \overline{HP} \Rightarrow M$ does not halt on $w \Rightarrow M^*$ accepts all odd-length strings and rejects all even-length strings $\Rightarrow L(M)$ contains all odd-length strings, i.e., is infinite, and $\overline{L(M)}$ contains all even-length strings, i.e., is infinite $\Rightarrow \langle M^*\rangle \in L_{26}$.
  $M \notin \overline{HP} \Rightarrow M$ halts on $w \Rightarrow M^*$ accepts all strings $\Rightarrow \overline{L(M)} = \emptyset$ i.e., $\overline{L(M)}$ is finite, and hence $\langle M^*\rangle \notin L_{26}$.

- $L_{27} = \{\langle M, x, k\rangle | M$ is a TM, and M does not halt on x within k steps$\}$.
  - **R.** Easy.
- $L_{28} = \{\langle M\rangle | M$ is a TM, and $|L(M)|$ is prime$\}$.
  - **Not RE.** Reduce $\overline{HP}$ as follows. $\tau(\langle M, w\rangle) = \langle M^*\rangle$. M^* on input x:
    if x is one of the first three strings (in lexicographic order) of Σ^*, then accept.
    Otherwise, run M on w. If M halts on w, and x is the 4^{th} string in Σ^* accept; otherwise, reject.

    Prove the validity (you will see that if M does not halt on w, then M^* accepts only 3 strings, which is a prime number of strings, and otherwise, it accepts 4 strings, which is not a prime number of strings. Formalize this!)
- $L_{29} = \{\langle M\rangle |$ there exists $x \in \Sigma^*$ such that for every $y \in L(M)$, $xy \notin L(M)\}$.

run m on x, and accept is m halts

  - **Not RE.** Reduce $\overline{HP}$ as follows. $\tau(\langle M, w\rangle) = \langle M^*\rangle$. M^* on input x:
    it runs M on w; if M halts on w, M^* accepts.
    If $\langle M, w\rangle \in \overline{HP}$ then $L(M^*)$ is empty, and hence $\langle M^*\rangle \in L_{29}$. WHY?
    If $\langle M, w\rangle \notin \overline{HP}$ then $L(M^*) = \Sigma^*$, and hence $\langle M^*\rangle \notin L_{29}$. WHY?

let x and y be the same strin
t, then l30 is simply
all turing machines

- $L_{30} = \{\langle M\rangle |$ there exist $x, y \in \Sigma^*$ such that either $x \in L(M)$ or $y \notin L(M)\}$.
  - **Recursive.** This is the language of all Turing machines!
  - **GOOD EXERCISE:** Solve the problem where we replace the "or" by "and".

every turing machine, we can come up with redundant , unreachable states

- $L_{31} = \{\langle M\rangle |$ there exists a TM M' such that $\langle M\rangle \neq \langle M'\rangle$ and $L(M) = L(M')\}$.
  - **Recursive.** This is the language of all Turing machines! (compare to L_{17}).
- $L_{32} = \{\langle M_1, M_2\rangle | L(M_1) \leq_m L(M_2)\}$.

this is talking about reduction
consider m1 to be a trivial turing machine that rejects everything
so we are essentially asking if m2 rejects everything as well => not re

  - **Not RE.** Reduction from $\overline{HP}$. $\tau(\langle M, w\rangle) = \langle M_1, M_2\rangle$.
    M_2 on input x: reject. (i.e., $L(M_2) = \emptyset$).
    M_1 on input y: run M on w; if M halts, check whether y is of the form $\langle M', z\rangle$, where M' is a TM, and z is a string. If so, run M' on z. If M' halts, M_1 accepts.
    You can now prove that: (1) If $\langle M, w\rangle \in \overline{HP}$ then $L(M_1) = \emptyset$ (in this case $L(M_1) \leq_m L(M_2)$ since $\emptyset \leq_m \emptyset$, and hence $\langle M_1, M_2\rangle \in L_{32}$), and (2) if $\langle M, w\rangle \notin \overline{HP}$ then $L(M_1) = HP$ (in which case $L(M_1) \not\leq_m L(M_2)$, since $HP \not\leq_m \emptyset$; why?).
    Therefore, $\overline{HP} \leq_m L_{32}$ and hence L_{32} is not in RE.
- $L_{33} = \{\langle M\rangle | M$ does not accept any string w such that 001 is a prefix of $w\}$.

usual niceu m'
m doesnt halt => m' doesnt accept anything => it belongs to l33
m does halt => m' accepts everything => it doesnt belong to l33

consider the case where x is epsilon, then this question
stands reduced to the following form : does m reject every string?
~HP is reducible to this => not RE

– **Not RE.** Simple reduction from $\overline{HP}$. $\tau(\langle M, w\rangle) = M^*$, where M^* on x: runs M on w; if M halts on w, M^* accepts.
Prove the validity of the reduction.

- $L_{34} = \{\langle M, x\rangle | M$ does not accept any string w such that x is a prefix of $w\}$.
  - **Not RE.** Proof similar to L_{33}.
- $L_{35} = \{\langle M, x\rangle | x$ is prefix of $\langle M\rangle\}$.
  - **Recursive.** The machine M^* that decides the language simply checks whether the string x is a prefix of the string $\langle M\rangle$.
- $L_{36} = \{\langle M_1, M_2, M_3\rangle | L(M_1) = L(M_2) \cup L(M_3)\}$.

consider the scenario where m2 and m3 are trivial turing machines rejecting all strings we are essentially asking if m1 rejects all strings => not RE

  - **Not RE.** Reduction from $\overline{HP}$. $\tau(\langle M, w\rangle) = \langle M_1, M_2, M_3\rangle$, where
    M_1 on x: runs M on w; if M halts, M_1 accepts.
    M_2 on y: reject. (i.e., $L(M_2) = \emptyset$).
    M_3 on z: reject. (i.e., $L(M_3) = \emptyset$).
    Prove the validity of the reduction.
- $L_{37} = \{\langle M_1, M_2, M_3\rangle | L(M_1) \subseteq L(M_2) \cup L(M_3)\}$.

not RE consider the case where m2 and m3 are trivial turing machines rejecting all strings => we are essentially asking if m1 rejects all strings as well => not re

  - **Not RE.** Reduction from $\overline{HP}$. $\tau(\langle M, w\rangle) = \langle M_1, M_2, M_3\rangle$, where
    M_1 on x: runs M on w; if M halts, M_1 accepts.
    M_2 on y: if y is the first string in Σ^* accept; else reject.
    M_3 on z: reject.
    Prove the validity of the reduction.
- $L_{38} = \{\langle M_1\rangle |$ there exist two TM's M_2 and M_3 such that $L(M_1) \subseteq L(M_2) \cup L(M_3)\}$.

trivially recursive since all turing machines will satisfy this criterion with m2=m1 and m3=m1

  - **Recursive.** This is the language of all Turing machines (simply take M_2 to be the machine that accepts everything, for example).
- $L_{39} = \{\langle M, w\rangle | M$ is a TM that accepts w using at most $2^{|w|}$ squares of its tape$\}$.
  - **Recursive.** Let m be the number of states in M, and k be the size of the alphabet that M uses, and $r = |w|$. If M uses at most 2^r squares of its tape, then there are at most $\alpha = mk^{2^r}2^r$ configurations (why?). If M runs on w for more than α steps, and does not use more than 2^r squares of its tape, then M must be in the one configuration at least twice (pigeonhole principle), in which case M would enter an infinite loop on input w.
    Based on this, we design a machine M^* that decides L_{39} that works as follows:
    M^* runs M on w for at most $\alpha + 1$ steps.
    If M accepts w within α steps with using at most 2^r squares, M^* halts and accepts.
    If M rejects w within α steps with using at most 2^r squares, M^* halts and rejects.
    If M goes beyond 2^r squares, M^* halts and rejects. If M does not halt and does not go beyond 2^r squares, M^* rejects.

no since the reduction of A to B is recursive, which makes A recursive

2. If $A \leq_m B$ and B is a regular language, does that imply that A is a regular language?
   NO! Take $A = \{a^n b^n : n \geq 0\}$, and $B = a^*$, and let the reduction from A to B be: if w is of the form $a^n b^n$, erase all the b's; otherwise, return b.

3. Recall the language $A_{TM} = \{\langle M, w\rangle | M$ is a TM, and M accepts $w\}$. Consider the language

$$J = \{w | w = 0x \text{ for some } x \in A_{TM} \text{ or } w = 1y \text{ for some } y \in \overline{A_{TM}}\}.$$

Using De Morgan's law,

$$\overline{J} = \{w | w = \varepsilon \text{ or } w = 1x \text{ for some } x \in A_{TM} \text{ or } w = 0y \text{ for some } y \in \overline{A_{TM}}\}.$$

(a) Show that J is not in RE.
Reduce $\overline{A_{TM}}$ to J: $\tau(w) = 1w$. Prove the validity of the reduction.

(b) Show that $\overline{J}$ is not in RE.
Reduce $\overline{A_{TM}}$ to $\overline{J}$: $\tau(w) = 0w$. Prove the validity of the reduction.

(c) Show that $J \le_m \overline{J}$.

BUT WHAT OF EPSILON?

$$\tau(w) = \begin{cases} 1x & if \quad w = 0x \\ 0x & if \quad w = 1x \end{cases}$$

Prove the validity of the reduction.

when A<=m B, ~A<=m ~B using the same map => ~A <=m A

4. Show that if a language A is in RE and $A \le_m \overline{A}$, then A is recursive.
Since $A \le_m \overline{A}$ it follows that $\overline{A} \le_m A$, and since A is in RE, it follows that $\overline{A}$ is also in RE. Since both A and $\overline{A}$ are in RE, it follows that A is in R (this follows from a theorem you learned in class).

5. A language L is **RE-Complete** if:

- $L \in RE$, and
- $L' \le_m L$ for all $L' \in RE$.

Recall the following languages:

$$L_{\Sigma^*} = \{\langle M \rangle |\ L(M) = \Sigma^*\}$$
$$HP = \{\langle M, w \rangle |\ M \text{ halts on } w\}$$

i have taken M' for M such that M' starts looping whenever M enters a reject state, so halting on w is the same as being accepted

(a) Is L_{Σ^*} RE-Complete or not? Prove your answer.
No. L_{Σ^*} is not in RE (Prove it by a reduction from $\overline{HP}$—similar to the reduction used with L_6). Hence, L_{Σ^*} is not RE-Complete.

(b) Is HP RE-Complete or not? Prove your answer.
Yes. You've seen in class that HP is in RE. We now prove that every language $L \in RE$ reduces to HP.
If L is in RE, then there exists a TM, M that semidecides it. The reduction from L to HP is as follows:
$\tau(w) = \langle M, w \rangle$, where M is the machine that semidecides L.
Prove the validity of the reduction. It's simple!

l2 can be a trivial language accepting all turing machines, like is l(m) = l(m) and l1 can be a trivial language rejecting all turing machines, like is l(m) != l(m). so not decidable

6. Let L_1, L_2 be two decidable languages, and let L be a language such that $L_1 \subseteq L \subseteq L_2$. Is L decidable or not? Prove your answer.
May be. Take $L_1 = \emptyset$ and $L_2 = \Sigma^*$, both of which are decidable. We know that $L_1 \subseteq HP \subseteq L_2$, and HP is not decidable. On the other hand, we know $L_1 \subseteq a^* \subseteq L_2$, and a^* is decidable.

7. Let L be a language RE. Show that $L' = \{x | \exists y : (x, y) \in L\}$ is also RE.
Assume M is a TM that semidecides L; we construct a TM, M', that semidecides L' (that uses M as a "black box"). Let the strings in Σ^* be $w_1, w_2, \ldots$. The TM M' on input x works as follows:

For $i = 1$ to "∞"

Run M for i steps on each of the strings $(x, w_1), (x, w_2), \ldots, (x, w_i)$.
If M accepts at least one of the strings, M' halts and accepts.

Prove that M' indeed semi-decides L'.

8. Prove or disprove: there exists an undecidable unary language (a unary language is a subset of 1^*).
There exists an undecidable unary language, since the number of unary languages is uncountable whereas the number of decidable languages is countably infinite! You need to know (understand and be able to prove) these facts.

9. PROBLEM FORMULATION.

(a) Consider the problem of testing whether a TM M on an input w ever attempts to move its head left when its head is on the leftmost tape cell. Formulate this problem as a language and show that it is undecidable.
The language is:

$$L = \{\langle M, w\rangle | M \text{ attempts to move its head left when its head is on the leftmost tape cell}\}.$$

We prove by contradiction that L is undecidable. Assume L was decidable, and let M^* be a TM that decides it. We construct a TM, M', that decides HP (the halting problem), i.e., we show HP is decidable as well, which is a contradiction.
The TM M' on input $\langle M, w\rangle$:

i. build a TM, M'', from M, where M'' shifts w one tape cell to the right, and mark the leftmost tape cell with $\sharp$.
ii. TM M'' runs M on w.
iii. If M'' encounters the $\sharp$ symbol during the run of M on w, then M'' moves to the right and simulates M reaching the leftmost tape cell.
iv. If M halts and accepts, M'' simulates moving its head all the way to the left and off the leftmost tape cell

The TM M' runs M^* on the input $\langle M', w\rangle$. If M^* accepts, M' accepts; otherwise, M' rejects. Notice that M^* always halts (accepts or rejects) since we assumed it is a decider.
You can prove now that M'' moves its head off of the leftmost tape cell iff M halts (and accepts) w. It then follows that M' decides HP, and hence HP is decidable, which is a contradiction. Therfore, the language L is not decidable.

(b) Consider the problem of testing whether a TM M on an input w ever attempts to move its head left at any point during its computation on w. Formulate this problem as a language and show that it is decidable.
The language is:

$$L = \{\langle M, w\rangle | M \text{ moves its head left on input } w\}.$$

The trick to proving this language is decidable is to notice that M moves its head left on an input w if and only if it does so within the first $|w| + |Q| + 1$ steps of its run on w, where $|Q|$ is the number of states of the machine M. Prove it!

after the first |w| steps, i have come to the side of blanks then i need to explore all possible states

Therefore, to decide whether an input $\langle M, w\rangle$ is an instance of L, the decider M^* simply runs M on w for at most $|w| + |Q| + 1$ steps and accepts iff M does moves its head left (the correctness of this construction follows immediately after you prove the validity of the above trick).

10. Let A and B be two disjoint languages. We say that language C **separates** A and B if $A \subseteq C$ and $B \subseteq \overline{C}$. Show that any two disjoint co-RE languages are separable by some decidable language.
$\overline{A}$ is in RE, and M_1 semi-decides it.
$\overline{B}$ is in RE, and M_2 semi-decides it.
We describe language C by a TM, M, that decides it:
M on input w:

run M_1 and M_2 on w (in interleaved mode).

If M_1 accepts, M rejects; If M_2 accepts, M accepts.
Notice that M always halts; we take $C = L(M)$.

11. Suppose there are four languages A, B, C, and D. Each of the languages may or may not be recursively enumerable. However, we know the following about them:

 - There is a reduction from A to B.
 - There is a reduction from B to C.
 - There is a reduction from D to C.

A <= B <= C
D <= C

Below are four statements. Indicate whether each one is

 (a) CERTAIN to be true, regardless of what problems A through D are.
 (b) MAYBE true, depending on what A through D are.
 (c) NEVER true, regardless of what A through D are.

Please, justify your answer!

c is atleast as difficult as a, so c can never be recursive

 (a) A is recursively enumerable but not recursive, and C is recursive.
 NEVER TRUE. reductions are transitive, and since A reduces to B, and B reduces to C, we conclude that A reduces to C. Therefore, if $A \notin R$, it can't be that $C \in R$.
 (b) A is not recursive, and D is not recursively enumerable. maybe true, then c will not be re
 MAYBE TRUE.

always true => d must always be recursive

 (c) If C is recursive, then the complement of D is recursive.
 CERTAIN to be TRUE. If C is in R, and since D reduces to C, then by the reduction theorem, it follows that D is in R. Since R is closed under complement, then the complement of D is also in R.
 (d) If C is recursively enumerable, then $B \cap D$ is recursively enumerable.
 CERTAIN to be TRUE. C is in RE implies that both B and D are in RE (by the reduction theorem), and so is their intersection (RE langs are closed under intersection).

12. Recall the following definition: A grammar G computes a function f iff for all $u, v \in \Sigma^*$,

$$SuS \Rightarrow_G^* v \text{ iff } f(u) = v.$$

Show a grammar that computes the following functions, defined on unary representations of natural numbers as follows:

- $f_1(n) = 3n + 5$.
  Solution: $S1 \to 111S$, $SS \to 11111$.
- $f_2(n) = \begin{cases} 1 & if \quad n \equiv 0\ (mod\ 3) \\ 11 & if \quad n \equiv 1\ (mod\ 3) \\ 111 & if \quad n \equiv 2\ (mod\ 3) \end{cases}$
  Solution: $111 \to \varepsilon$, $SS \to 1$, $S1S \to 11$, $S11S \to 111$.
- $f_3(n) = n - 1$.
  $S1 \to \varepsilon$, $1S \to 1$.
- $f_4(n) = n/2$. Assume n is even.
  $S11 \to 1S$, $SS \to \varepsilon$.
- $f_5(w) = ww$, where $w \in \{a, b\}^*$.
  The solution is in the course packet.
- $f_6 = w'$, where $w \in \{a, b\}^*$, and w' is obtained from w by replacing the a's by b's and b's by a's. For example, $f_6(aaba) = bbab)$.
  $Sa \to bS$, $Sb \to aS$, $SS \to \varepsilon$.
- $f_7(a_1a_2 \ldots a_k) = a_1a_1a_2a_2 \ldots a_ka_k$, where each a_i is in the alphabet $\{a, b\}$. For example, $f_7(aaba) = aaaabbaa$.

  $Sa \to aaS$, $Sb \to bbS$, $SS \to \varepsilon$.
- $f_8(w) = \begin{cases} f_6(w) & if \quad the\ rightmost\ symbol\ of\ w\ is\ a \\ f_7(w) & if \quad the\ rightmost\ symbol\ of\ w\ is\ b \end{cases}$ $(\Sigma = \{a, b\})$.
  $aS \to Tb$, $bS \to Ubb$,
  $aT \to Tb$, $bT \to Ta$, $ST \to \varepsilon$
  $aU \to Uaa$, $bU \to Ubb$, $SU \to \varepsilon$.

Text

13. Show that the following languages are recursive.
I provide only solution sketch; please make sure you write the solutions formally, in terms of TMs that decide the languages!

- $L_{40} = \{\langle M\rangle | M$ is a DFA and $L(M)$ is finite$\}$.
  Check whether there is a loop in M that is reachable from the start state.
- $L_{41} = \{\langle M\rangle | M$ is a DFA and $L(M) = \Sigma^*\}$.
  Minimize the DFA and check whether it had only one state with a self-loop on all the alphabet symbols.
- $L_{42} = \{\langle M, x\rangle | M$ is a DFA and M accepts $x\}$.
  Run M on x.
- $L_{43} = \{\langle M, x\rangle | M$ is a DFA and M halts on $x\}$.
  DFA's always halt!!!
- $L_{44} = \{\langle G\rangle | G$ is a CFG and $L(G) = \emptyset\}$.
  Check whether any terminal string can be generated by G (you can convert into CNF first).
- $L_{45} = \{\langle M\rangle | M$ is a DFA and M accepts some string of the form ww^R for some $w \in \{a, b\}^*\}$.
  Intersect the DFA with a PDA that recognizes the language $\{ww^R | w \in \{a, b\}^*\}$, and test the emptiness of the resulting PDA. (You learned in class how to build the intersection of a PDA and a DFA).

for all states, construct two DFAs, one having the start state
as final state, and the other having the final state as final state
and check if the intersection of the two languages
is empty

14. Prove that each of the following languages are not context-free, and write unrestricted grammars that generate them.

- $L_{46} = \{x\sharp w|\ x, w \in \{a,b\}^*$ and x is a substring of $w\}$.
Use the PL to prove the language is not CF. An unrestricted grammar that generates it:
$S \to S_1XT$
$S_1 \to aS_1a|bS_1b|\sharp TY$
$T \to aT|bT|\varepsilon$
$Y \to YA$
$Aaa \to aAa$
$Abb \to bAb$
$Aba \to aAb$
$Aab \to bAa$
$AaX \to Xa$
$AbX \to Xb$
$YX \to \varepsilon.$

consider 1^k#1^k
if # selected => violation
if x selected => violation
if w selected => violation(for i = 0)

- $L_{47} = \{w \in \{a,b,c\}^*|\ \sharp_a(w) \geq \sharp_b(w) \geq \sharp_c(w)\}$.
Use the PL to prove the language is not CF. An unrestricted grammar that generates it:
a^kb^kc^k
$S \to ABSC|ABS|T$
$T \to AT|\varepsilon$
$AB \to BA$
$BA \to AB$
$BC \to CB$
$CB \to BC$
$AC \to CA$
$CA \to AC$
$A \to a$
$B \to b$
$C \to c.$

- $L_{48} = \{a^nb^nca^nb^n|n > 0\}$.
the string pumps itself lmaooo
$S \to ABSXY$, $S \to ABcXY$
$YX \to XY$, $BA \to AB$
$Bc \to bc$, $Bb \to bb$
$Ab \to ab$, $Aa \to aa$
$cX \to ca$, $aX \to aa$
$aY \to ab$, $bY \to bb$

- $L_{49} = \{a^nb^{2n}c^{3n}|n \geq 0\}$.
the string pumps itself lmaooo
$S \to aBSccc|\varepsilon$
$Ba \to aB$
$Bc \to bbc$
$Bb \to bbb.$

- $L_{50} = \{a^nb^{n+m}c^md^n|m, n \geq 0\}$.
a^kb^2kc^kd^k
$S \to aBSd|T$
$T \to bTc|\varepsilon$

$Ba \rightarrow aB$
$Bb \rightarrow bb$
$Bd \rightarrow bd.$

- $L_{51} = \{w \in \{1\}^* | w$ is the unary encoding of 2^k for some $k > 0\}$.
  $S \rightarrow L1R$
  $L \rightarrow LL$
  $L1 \rightarrow 11L$
  $LR \rightarrow R$
  $R \rightarrow \varepsilon.$

15. Let L_{52} be the language containing only the single string s, where

$$s = \begin{cases} 0 & if \quad God\ does\ not\ exist \\ 1 & if \quad God\ exists \end{cases}$$

Is L_{52} decidable? Why or why not? (Note that the answer does not depend on your religious convictions.)

L_{52} is recursive. $A = \{0\}$ or $A = \{1\}$, and for each possibility there is a TM that decides it!