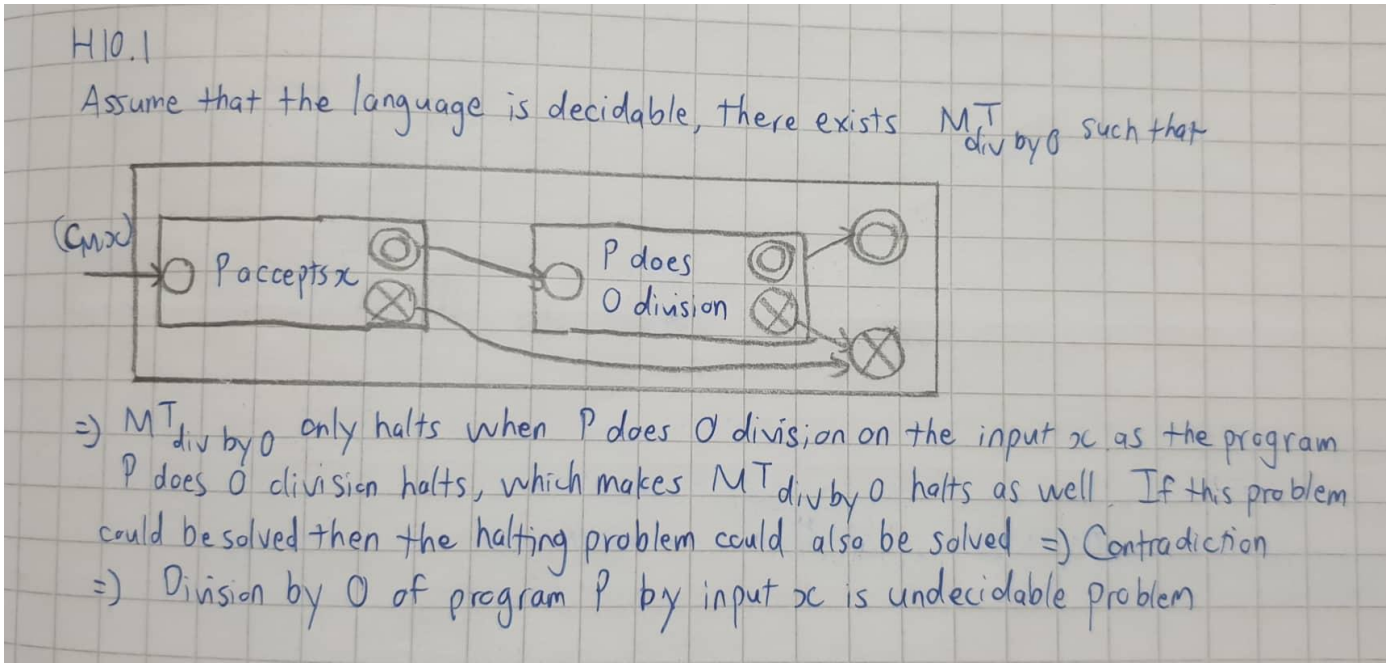
Homework Problems

H10.1 Division by zero is a common error in computer programs. Naturally, it would be very good to make sure that our software is free of such errors before it is published. Thus, let us consider the following decision problem DIV-BY-ZERO:

> Given a computer program P (in your favourite programming language) that can use an unbounded amount of memory. Is there any input that causes the program to perform division by zero?

Explain why this problem is undecidable. (Assuming that an unbounded amount of memory is really accessible, i.e. pointers are not restricted to some fixed bitlength etc.)

Hint: Show that if the problem were decidable, then so would be the Halting Problem for Turing machines. That is, describe how one can transform a pair $\langle M, x \rangle$, where M is a Turing machine and x is an input string, into a computer program P such that M halts on x if and only if P performs division-by-zero on some input. You don't have to give full details of the transformation, a high-level description is enough.



Proof: For every program:

- Replace every HALT command by a Division by Zero.
- Replace every division by;
  - -- A Test to Determine if the Denominator is Zero
  - -- If the Test is Positive:
    - -- Perform an action equivalent to Division by Zero
    - -- Jump around the Division

    Otherwise just Perform the Division
- New Programs Divides by Zero $\Leftrightarrow$ Old Program HALTS
- If we could solve the Division by Zero Problem we could also solve the HALTING Problem.
- Therefore we cannot solve the Division by Zero Problem.

Proving set S is undecidable

1. **Assume that there is some (hypothetical) program that decides S**
2. **Choose some known undecidable set K**
3. **Show how to build a program to decide K that uses the program for S as a subroutine**
   - **Often you only need one call to S and can return the same answer**
4. **Prove that your algorithm outputs true on its input if and only if that input is in K, using the correctness of the hypothetical program for S**
   - **If you made one call and returned the same answer you just need to show that the input to the program is in K iff the value in the call to the subroutine is in S.**
5. **"Since K is undecidable, the program deciding S can't exist. Therefore S is undecidable."**

H10.2 Prove that the following problem is undecidable:

Given a Turing machine M, does it accept the empty input string ε?

Hint: Consider the proof of Lemma 10.5 or the examples following Corollary 10.9 on the lecture slides.

Call $L_1 = \{\langle M \rangle \mid M \text{ accepts } \lambda\}$ and $L_2 = \{\langle M \rangle \mid M \text{ accepts only } \lambda\}$, where $\langle M \rangle$ denotes the description of the TM M. We wish to establish a mapping reduction $L_1 \leq_{\mathrm{M}} L_2$.

To do this, we want a mapping, f, from TM descriptions to TM descriptions such that

$$\langle M \rangle \in L_1 \iff f(\langle M \rangle) \in L_2$$

For a TM description $\langle M \rangle$, we'll define $f(\langle M \rangle) = \langle N \rangle$ where

```
N(x) =
   if x = lambda
      run M on input lambda
      if M accepts
         accept
```

Now what happens with this mapping?

- If $M(\lambda)$ accepts (so $\langle m \rangle \in L_1$), N will accept λ and no other string, so $\langle N \rangle \in L_2$.
- If $M(\lambda)$ doesn't accept (so $\langle M \rangle \notin L_1$), N will accept no input string, so $L(N) = \varnothing$ and thus $\langle N \rangle \notin L_2$.

That's exactly what we need to establish the reduction $L_1 \leq_{\mathrm{M}} L_2$.

From Collorary 10.9, If A $\leq_{\mathrm{m}}$ B and A is not decidable, then B is not decidable. Therefore whether the given Turing machine accepts empty input string is undecidable.

H10.3 Consider the following languages. For each language: (i) give a string that belongs to the language, (ii) give a string that does not belong to the language, (iii) determine whether the language is decidable or not. Justify your answers.

(a) $\{x \in \{0,1,...,9\}^* \mid x \text{ is (a decimal representation of) a prime number}\}$

(b) $\{\langle l, m\rangle \mid l \text{ is a list of rational numbers whose average is less than the rational number } m\}$

(c) $\{p \mid p \text{ is a polynomial on } n \text{ variables that has integer zeros (roots) for some values of its variables in the domain } [-1000000, 1000000]^n\}$

(d) $\{\langle G, x\rangle \mid G \text{ is a context-free grammar that generates string } x\}$

(e) $\{\langle G, s, t\rangle \mid G \text{ is a finite undirected graph that has a path from vertex } s \text{ to vertex } t\}$

(f) $\{\langle G, k\rangle \mid G \text{ is a finite undirected graph that contains a clique of } k \text{ or more vertices}\}$[1]

(g) $\{P \mid P \text{ is a Python program that does not contain any division operators}\}$

(h) $\{P \mid P \text{ is a Python program whose execution halts eventually when it is run without any input and has unbounded amount of memory available}\}$

Note: For items (b)–(h), where no specific alphabet or representation of the language elements as strings is given, you may choose any reasonable encoding. For instance, Python programs can be viewed as strings over the Unicode alphabet. Similarly, polynomials can be expressed in the ASCII alphabet so that for instance the polynomial $7x_1x_2^3 + 1.5x_1^7$ is expressed as the string `7*x1*x2^3+1.5*x1^7`.

(a)
(i) A string that belongs to the language: 17
(ii) A string that does not belong to the language: 16
(iii) Decidable or Undecidable: We know that there is a deterministic program that can determine whether a number is a prime or not. Now for this language, it can simply convert string representation of x into an integer. If x is anything other than an integer, the program halts immediately. Otherwise it would check if it is prime or not and then halts

(b)
(i) A string that belongs to the language: <(½, ¼), 1>
(ii) A string that does not belong to the language <(5/3, 7/4), 1>

(iii) Decidable or Undecidable: Decidable. This language can detect deterministically which are the symbols /, comma and <> symbols, so it can find out the average of rationals and the value of m. Finally it compares and then the Turing machine that recognizes this language always halt

(c)
(i) A string that belongs to the language: n^2 - 4 = 0
(ii) A string that does not belong to the language: n^2 - 3 = 0
(iii) Decidable or Undecidable: Undecidable. There is no algorithm that completely, and in finite time can tell us whether a multivariate polynomial has an integer root. Since there are no such algorithms, this language is undecidable

(d)
(i) A string that belongs to the language:
<S -> AB
A -> a
B -> b
, ab>
(ii) A string that does not belong to the language
<S -> BA
A -> a
B -> b
, ab>
(iii) Decidable or Undecidable: Decidable. The CFG can be converted to Chomsky's Normal Form and the parse strings of this CNF will always determine if the input belongs to the parse tree or not

(e)
(i) A string that belongs to the language:
< (a - b
b - c),
a, c>
(ii) A string that does not belong to the language
< (a - b
b - d,
c - c),
a, c >
(iii) Decidable or Undecidable: Decidable. From vertex s, we can perform breadth first search and if it eventually meets t, then the program returns yes. After spanning all possible ways (excluding cyclic paths) and t is still not one of the visited vertex => returns no

(f)
(i) A string that belongs to the language:
< (a - b
b - c,

c - a),
3 >
(ii) A string that does not belong to the language:
< (a - b
b - c,
c - a),
2 >
(iii) Decidable or Undecidable: Decidable. The idea is to use recursion to solve the above problem. All the vertices whose degree is greater than or equal to (K-1) are found and checked which subset of K vertices form a clique. When another edge is added to the present list, it is checked if by adding that edge, the list still forms a clique or not.

(g)
(i) A string that belongs to the language: print("Hello World")
(ii) A string that does not belong to the language: print(5/3)
(iii) Decidable or Undecidable: Decidable: the program can just scan from start till the end. If it contains "/" then it returns no, otherwise yes

(h)
(i) A string that belongs to the language:

```
i = 0
while (i < 1000000000):
    i +=1
```

(ii) A string that does not belong to the language:

```
i = 0
while (true):
    i +=1
```

(iii) Decidable or Undecidable: Undecidable. If this problem is solvable, then so is the Halting problem, which is not possible => This problem is undecidable as well