Lecture #8: Proving Undecidability — Part One

What To Do Before the Lecture

1. Watch the videos for Lecture #8 —noting that they will probably be understandable if you play them at double speed. If you do not have time for this then look at the "Key Concepts" document that is found, immediately after the videos for this lecture, on the course web site, instead.
2. ***Print*** and read through the rest of this document and — if you have time — try to solve the problems! These should help you to check that you understand how Turing machines process strings, and that you understand how to prove things about them.
3. The supplemental document is for interest only. It includes an outline of a proof that the language L_{TM} and $L_{\mathsf{TM+I}}$ are decidable.

Problems To Be Solved

Proving That A_{TM} is Undecidable

Claim. *The language A_{TM} is undecidable.*

How To Prove This:

On input $\mu \in \Sigma^{\star}_{\mathsf{TM}}$:

1. `if` $(\mu \in L_{\mathsf{TM}})$ `{`
   Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{accept}}, q_{\mathsf{reject}})$ be the Turing machine that is encoded by μ.
2. `if` $(\Sigma = \Sigma_{\mathsf{TM}})$ `{`
3. `if` (M accepts μ) `{` // *Does M accepts its own encoding?*
4. `reject` μ
   `}` `else` `{`
5. `accept` μ
   `}`
   `}` `else` `}`
6. `reject` μ
   `}`
   `}` `else` `{`
7. `reject` μ
   `}`

Figure 1: Algorithm Used to Get a Contradiction

Details of Proof:

Suppose, to obtain a contradiction, that the language A_{TM} is decidable. Consider the algorithm shown in Figure 1.

Subclaim: There exists a Turing machine M_D that implements the algorithm in Figure 1.

Explanation:

It follows that there exists a string $\mu_D \in \Sigma^{\star}_{\mathsf{TM}}$ that encodes the Turing machine M_D.

What Happens When M_D is Executed on its Encoding, μ_D?

An "Oracle Reduction" Problem That is Suitable for a Test

Let $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$, let $L \subseteq \Sigma^\star$, and let

$$L_a = \{\omega \in \Sigma^\star \mid \omega \in L \text{ and } \omega \text{ ends with "a"}\}.$$

You were asked to prove that $L_a \preceq_{\mathsf{O}} L$.

What Do You Need To Provide?

```
On input ω ∈ {a, b, c}*:
1. if (ω ends with "a") {
2.   if (ω ∈ L) {
3.     accept ω
     } else {
4.     reject ω
     }
   } else {
5.   reject ω
   }
```

Figure 2: Algorithm Used for an Oracle Reduction

Details of Reduction

Consider the algorithm shown in Figure 2.

Establishing Correctness

Implementation Details

Why is This “Suitable as a Test Problem”?

An "Oracle Reduction" Problem That is Suitable for an Assignment

Consider the language $\text{LOOP}_{\text{TM}} \subseteq \Sigma^{\star}_{\text{TM}}$, including encodings of Turing machines M and input strings ω for M such that M ***loops*** on ω.

You were asked to establish an oracle reduction, involving LOOP_{TM} and some other language, that can be used to prove that the language LOOP_{TM} is undecidable.

What We Need To Provide:

Useful Information That We Already Have:

At this point in the course several similar languages have been considered:

- The language $L_{\text{TM+I}} \subseteq \Sigma^{\star}_{\text{TM}}$ of encodings of Turing machines M and input strings ω for M. This language is ***decidable*** and it follows from the definitions of these languages that $\text{LOOP}_{\text{TM}} \subseteq L_{\text{TM+I}} \subseteq \Sigma_{\text{TM}}$.
- The language $A_{\text{TM}} \subseteq \Sigma^{\star}_{\text{TM}}$ of encodings of Turing machines M and input strings ω for M such that M accepts ω.

  This language is ***recognizable***: A multi-tape Turing machine with language A_{TM} (called a "universal Turing machine") was described in Lecture #8 — and it follows, by results about multi-tape Turing machines included in Lecture #7, that there must also exist a standard (single tape) Turing machine, M_{UTM}, whose language is A_{TM}, as well.

  On the other hand it was proved, above, that the language A_{TM} is ***undecidable***.

Which Oracle Reduction Will We Present, To Solve This Problem?

Details of Reduction: An Algorithm To Consider

Details of Reduction: Implementation Details

Why is This More Appropriate for an Assignment?

Proving that A_{TM}^{C} is Unrecognizable — If Time Permits

Claim. *The language A_{TM}^{C} is unrecognizable.*

How To Prove This:

Details of Proof:

Suppose to obtain a contradiction, that A_{TM}^{C} is recognizable.

- Since A_{TM} is recognizable (because it is the language of a universal Turing machine) there exists a (standard) Turing machine M_{Yes} such that $L(M) = A_{\mathsf{TM}}$.
- Since A_{TM}^{C} is also recognizable (by the above assumption) there is also a (standard) Turing machine M_{No} such that $L(M) = A_{\mathsf{TM}}^{C}$.

```
On input μ ∈ Σ*_TM:
 1. integer i := 1
 2. while (true) {
 3.   if (M_Yes accepts μ after taking at most i steps) {
 4.     accept μ
 5.   } else if (M_Yes rejects μ after taking at most i steps) {
 6.     reject μ
      }
 7.   if (M_No accepts μ after taking at most i steps) {
 8.     reject μ
 9.   } else if (M_No rejects μ after taking at most i steps) {
10.     accept μ
      }
11.   i := i + 1
    }
```

Figure 3: Another Algorithm Used to Get a Contradiction

Consider the algorithm shown in Figure 3.

What Happens When This Algorithm is Executed on μ, when $\mu \in A_{\mathsf{TM}}$?

What Happens When This Algorithm is Executed on μ, when $\mu \notin A_{\mathsf{TM}}$?

What Can We Conclude From This?