CSC363 Tutorial #4

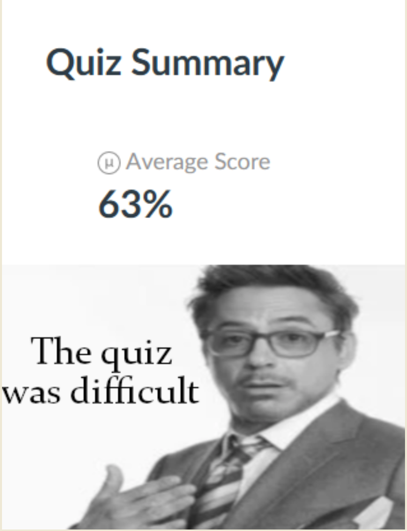
Halting problem

February 8, 2023

Things covered in this tutorial

- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ How was I supposed to do Quiz 2?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?
- ⋆ What's the halting problem?

Quiz 2 solutions?



Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e.[1] sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i.

Which of the following is true about the union $U = \bigcup_{i=0}^{\infty} A_i$?[2]

- ⋆ U is c.e..
- ⋆ U is computable.
- ⋆ U is not c.e..

[1] Recall that c.e. is a synonym for recognizable.

[2] For those unfamiliar with this $\bigcup$ notation: $x \in U$ if and only if $x \in A_i$ for some $i \in 0, 1, \ldots$.

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e.[1] sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i.

Which of the following is true about the union $U = \bigcup_{i=0}^{\infty} A_i$?[2]

- U is c.e..
- U is computable.
- U is not c.e..

[1] Recall that c.e. is a synonym for recognizable.

[2] For those unfamiliar with this $\bigcup$ notation: $x \in U$ if and only if $x \in A_i$ for some $i \in 0, 1, \ldots$.

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

Proof. Define the recognizer T for U as follows:

```
T(x):
    i = 0
    while True:
        run M(i)
        if M(i) outputs x:
            accept
        i += 1
```

Question: What is wrong with the above proof?

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

Proof 2 (better!). Define the recognizer T for U as follows:

```
T(x):
    i = 0
    while True:
        run M(0), M(1), ..., M(i) for i steps each
        if any of the above output x:
            accept
        i += 1
```

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

U might not be computable!

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that there exists a machine M which, when given i as an input, M starts outputting the elements of A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. Then U is c.e..

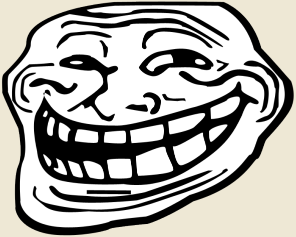
U might not be computable! Take
$A_0 = A_1 = A_2 = \ldots =$ The halting problem. Then
$U =$ The halting problem as well, which is not computable.

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that for each i, there exists an enumerator M_i for A_i.

Which of the following is true about the union $U = \bigcup_{i=0}^{\infty} A_i$?

- $\star$ U is c.e..
- $\star$ U is computable.
- $\star$ U might not be c.e..



Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that for each i, there exists an enumerator M_i for A_i.

Which of the following is true about the union $U = \bigcup_{i=0}^{\infty} A_i$?

- ⋆ U is c.e..
- ⋆ U is computable.
- ⋆ U might not be c.e..

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that for each i, there exists an enumerator M_i for A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. U might not be c.e..

Proof. Let S be any non-c.e. set. Define

$$A_i = \begin{cases} \{i\} & i \in S \\ \emptyset & i \notin S \end{cases}$$

Question: What is $\bigcup_{i=0}^{\infty} A_i$?

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that for each i, there exists an enumerator M_i for A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. U might not be c.e..

Proof. Let S be any non-c.e. set. Define

$$A_i = \begin{cases} \{i\} & i \in S \\ \emptyset & i \notin S \end{cases}$$

Question: What is $\bigcup_{i=0}^{\infty} A_i$? **Answer:** S!

Pop quiz time!

Suppose $A_0, A_1, A_2, \ldots$ is a countably infinite collection of c.e. sets. Also suppose that for each i, there exists an enumerator M_i for A_i. Define $U = \bigcup_{i=0}^{\infty} A_i$. U might not be c.e..

Proof. Let S be any non-c.e. set. Define

$$A_i = \begin{cases} \{i\} & i \in S \\ \emptyset & i \notin S \end{cases}$$

Question: What is $\bigcup_{i=0}^{\infty} A_i$? **Answer:** S!

Takeaway: You are given that *there exists* an enumerator M_i for each A_i. However, that does not necessarily mean that you can **computably** construct M_i, given i.

Pop quiz time!

It is possible to have a set which is both computable and c.e..

- ⋆ True.
- ⋆ False.

Pop quiz time!

It is possible to have a set which is both computable and c.e..

- ⋆ True.
- ⋆ False.

All computable sets are c.e.!

Halt or loop?

Question: For which input $x \in \mathbb{N}$ does the following function halt?

```
def f(x):
    while x != 0:
        x += 1
    return x
```

Halt or loop?

Question: For which input $x \in \mathbb{N}$ does the following function halt?

```
def f(x):
    while x != 0:
        x += 1
    return x
```

Answer: $f(x)$ only halts for $x = 0$.

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    if x = 0:
        return y + 1
    else if y = 0:
        return f(x - 1, y + 1)
    else:
        return f(x - 1, f(x, y - 1))
```

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    if x = 0:
        return y + 1
    else if y = 0:
        return f(x - 1, y + 1)
    else:
        return f(x - 1, f(x, y - 1))
```

Answer: This is the Ackermann function, which halts for all $x, y \in \mathbb{N}$ (but takes a very long time!)

You can try running `ackermann.py`.

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    n = 100
    while True:
        M_x = the x-th Turing Machine
        run M_x(y) for n steps
        if it halted within n steps:
            return
        n += 1
```

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    n = 100
    while True:
        M_x = the x-th Turing Machine
        run M_x(y) for n steps
        if it halted within n steps:
            return
        n += 1
```

Answer: f halts on inputs $\{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$.

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    n = 0
    while True:
        M_(x + n) = the (x + n)-th Turing Machine
        run M_(x + n)(y) for n steps
        if it halted within n steps:
            return
        n += 1
```

Halt or loop?

Question: For which inputs $x, y \in \mathbb{N}$ does the following function halt?

```
def f(x, y):
    n = 0
    while True:
        M_(x + n) = the (x + n)-th Turing Machine
        run M_(x + n)(y) for n steps
        if it halted within n steps:
            return
        n += 1
```

Answer: f halts on all inputs x, y![3]

[3] Let $S = \{e : M_e(y) \text{ halts}\}$. S is an infinite set (why?). Thus there is some $n \in \mathbb{N}$ for which $x + n \in S$ (why?).

Halt or loop?

Question: For which inputs $x \in \mathbb{N}$ does the following function halt?

```
def f(x):
    while x != 1:
        if x is odd:
            x = 3x + 1
        else:
            x = x / 2
```

Halt or loop?

Question: For which inputs $x \in \mathbb{N}$ does the following function halt?

```
def f(x):
    while x != 1:
        if x is odd:
            x = 3x + 1
        else:
            x = x / 2
```

Answer: We don't know... This is the unsolved **Collatz Conjecture** in mathematics. Try running `collatz.py`!

Halting problem

From lecture:

$$\mathrm{HP} = \{x \in \mathbb{N} : M_x(x) \text{ halts}\}$$

is an undecidable language!

Halting problem

From lecture:

$$\mathrm{HP} = \{x \in \mathbb{N} : M_x(x) \text{ halts}\}$$

is an undecidable language!

There is no algorithm that determines whether a given program P halts, when P is given its own source code as the input.

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 1. Suppose, towards a contradiction, there was a decider D for HP_2.

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 1. Suppose, towards a contradiction, there was a decider D for HP_2. Build a Turing machine M as follows:

```
M(x):
    run D(x, x)
    if D accepts:
        loop
    else:
        accept
```

Suppose M is the e-th Turing machine.
Question: Does $M(e)$ halt or loop?

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 1. Suppose, towards a contradiction, there was a decider D for HP_2. Build a Turing machine M as follows:

```
M(x):
    run D(x, x)
    if D accepts:
        loop
    else:
        accept
```

Suppose M is the e-th Turing machine.
Question: Does $M(e)$ halt or loop? **Answer (click me!)**

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 2. Suppose, towards a contradiction, there was a decider D for HP_2.

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 2. Suppose, towards a contradiction, there was a decider D for HP_2. Build a Turing machine M as follows:

```
M(x):
    return D(x, x)
```

This contradicts the fact that

$$\mathrm{HP} = \{x \in \mathbb{N} : M_x(x) \text{ halts}\}$$

is an undecidable language.

Halting problem

Another version of the halting problem:

$$\mathrm{HP}_2 = \{(x, y) \in \mathbb{N}^2 : M_x(y) \text{ halts}\}$$

is an undecidable language!

Proof 2. Suppose, towards a contradiction, there was a decider D for HP_2. Build a Turing machine M as follows:

```
M(x):
    return D(x, x)
```

This contradicts the fact that

$$\mathrm{HP} = \{x \in \mathbb{N} : M_x(x) \text{ halts}\}$$

is an undecidable language.

Such an argument of "if you can decide one language, then you can decide another language" is called a reduction.