CSCI 112: Quines and the Halting Problem

Geoffrey Matthews

March 31, 2023

The Halting Problem

- Write a program that checks other programs for infinite loops.
- For example, with the input:

```
while (True):
    continue
```

  your program will output: `Infinite loop!`

- But with the input:

```
print(2+2)
```

  your program will output: `Halts!`

- Your program must halt in every case, and so you can't just run the program in a simulator and check. You might wait forever.
- Is there another way it could be done?

The Halting Problem

- Write a program that checks other programs for infinite loops.
- For example, with the input:

```
while (True):
    continue
```

  your program will output: `Infinite loop!`

- But with the input:

```
print(2+2)
```

  your program will output: `Halts!`

- Your program must halt in every case, and so you can't just run the program in a simulator and check. You might wait forever.
- Is there another way it could be done?

The Halting Problem

- Write a program that checks other programs for infinite loops.
- For example, with the input:

```
while (True):
    continue
```

  your program will output: `Infinite loop!`

- But with the input:

```
print(2+2)
```

  your program will output: `Halts!`

- Your program must halt in every case, and so you can't just run the program in a simulator and check. You might wait forever.
- Is there another way it could be done?

Paradoxical sentences

This sentence is false.

- If it's true, then it's false.
- If it's false, then it's true.
- Is self-reference the problem?

Paradoxical sentences

This sentence is false.

- If it's true, then it's false.
- If it's false, then it's true.
- Is self-reference the problem?

Paradoxical sentences

This sentence is false.

- If it's true, then it's false.
- If it's false, then it's true.
- Is self-reference the problem?

Paradoxical sentences

This sentence is false.

- If it's true, then it's false.
- If it's false, then it's true.
- Is self-reference the problem?

Sentences can talk about other sentences

- **"I love you!"** is what I want to hear.
- **"Cats can fly."** is false.
- **"my mother hates the"** is a sentence fragment.
- Appending **"Cats cannot"** and **"fly"** yields a truth.

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- This sentence manages to talk about itself without using self-reference.
- It only refers to the quoted sentence fragment, not itself.
- The sentence says something about what happens when you do something to the sentence fragment.
- When you do what the sentence says, you get the sentence itself.
- Is this sentence true or false?
  - If it's true, then it's false.
  - If it's false, then it's true.

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- This sentence manages to talk about itself without using self-reference.
- It only refers to the quoted sentence fragment, not itself.
- The sentence says something about what happens when you do something to the sentence fragment.
- When you do what the sentence says, you get the sentence itself.
- Is this sentence true or false?
  - If it's true, then it's false.
  - If it's false, then it's true.

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- This sentence manages to talk about itself without using self-reference.
- It only refers to the quoted sentence fragment, not itself.
- The sentence says something about what happens when you do something to the sentence fragment.
- When you do what the sentence says, you get the sentence itself.
- Is this sentence true or false?
  - If it's true, then it's false.
  - If it's false, then it's true.

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- This sentence manages to talk about itself without using self-reference.
- It only refers to the quoted sentence fragment, not itself.
- The sentence says something about what happens when you do something to the sentence fragment.
- When you do what the sentence says, you get the sentence itself.
- Is this sentence true or false?
  - If it's true, then it's false.
  - If it's false, then it's true.

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- The beauty of the Quine sentence is that it contains a *recipe*, or a *set of instructions*, that, when carried out, result in a paradoxical situation.
- Computers can also talk about what happens when you carry out instructions!

The Quine sentence

"yields falsehood when appended to its own quotation" yields falsehood when appended to its own quotation.

- The beauty of the Quine sentence is that it contains a *recipe*, or a *set of instructions*, that, when carried out, result in a paradoxical situation.
- Computers can also talk about what happens when you carry out instructions!

Programs can analyze or even run other programs

- A syntax checker for *Python* can be written in *Python*.
  - This program could check its own syntax!
- An interpreter for *Python* can be written in *Python*.
  - This program could interpret itself!

Programs analyzing themselves

- But when we run a program on itself, we assume that, while the program is running, there is another copy of the program stored somewhere.
- Even this is not necessary! We can use Quine's trick!
- Any program can be modified so that, before it does it's usual job, it prints a copy of its own source code.
- Such a program is called a *Quine*.

Python Quines

- Consider first this program. Its output is below.

```
#data = """Some random
text of whatever form.
Bla bla bla."""
def printDataAsData(data):
    print('data = """' + '"' + data + '"""' + '"')
def printDataAsProgram(data):
    print(data)
printDataAsData(data)
printDataAsProgram(data)
print(2+2)
```

```
data = """Some random
text of whatever form.
Bla bla bla."""
Some random
text of whatever form.
Bla bla bla.
4
```

Scheme Quines

- Clearly we could replace the text string with anything—even the remaining text of the program.

```
data = """def printDataAsData(data):
    print('data = """ + '"' + data + '"""' + '"')
def printDataAsProgram(data):
    print(data)
printDataAsData(data)
printDataAsProgram(data)
print(2+2)"""
def printDataAsData(data):
    print('data = """ + '"' + data + '"""' + '"')
def printDataAsProgram(data):
    print(data)
printDataAsData(data)
printDataAsProgram(data)
print(2+2)
```

- Why didn't I show you the output of this program?

Quines

- Clearly adding two and two could be replaced with *any* program.
- It is thus easy to see that *any* program could be modified to print its own source code before doing anything else.
- This can be done in any language:
  `https://www.nyx.net/~gthompso/quine.htm`
- Thus, any program that analyzes other programs, *can also analyze itself!*
- We can use this trick to make some paradoxical looking programs.

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

The Halting Problem Again

- Suppose the halting problem is solvable, and program P can analyze any program to determine if it halts. Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy runs forever, then halt.
  - Else loop forever.
- What does P say about Q?
  - If P says Q halts, then Q runs forever.
  - If P says Q runs forever, then Q halts.

The halting problem is not solvable!

All problems about computer programs are unsolvable!

- Suppose program P can analyze any program to determine if it prints "hello". Program P always halts.
- Now we build a program Q, which uses P as a module.
  - Program Q first obtains a copy of itself, Q'.
  - Program Q now runs P on Q'.
  - If P says that the copy prints "hello", then print "goodbye".
  - Else print "hello".
- What does P say about Q?
  - If P says Q prints "hello", then Q prints "goodbye".
  - If P says Q does not print "hello", then Q prints "hello".

No computer program can determine anything interesting about computer programs!

—Rice's Theorem, 1951