Today’s Advice

- My only piece of advice for today:
- Don’t listen to anything I say today; it will hurt your head.
Self Contradiction

• The main topic today is self reference and self contradiction.

• The idea is that “interesting” things happen when something can refer to itself and assert that it has properties that negate its own existence.

Oxymorons

• We’re all familiar with oxymorons: words that harbor two conflicting meanings.

• Top ten list from http://www.oxymoronlist.com/:

  18. Personal Computer    8. Working Vacation
  17. Silent Scream           7. Tax Return
  16. Living Dead            6. Virtual Reality
  15. Same Difference        5. Dodge Ram
  12. Tight Slacks           2. Healthy Tan
  11. Peace Force            1. Microsoft Works
Surface Contradiction

• Examples seem incongruous, but they all actually make sense.
  • “jumbo shrimp” just means pretty big for a shrimp, which makes perfect sense. Like “pretty fly for a white guy”.
  • The contradiction isn’t very deep.

You Don’t Say...

• I always say, "Silence is golden."
• We’re not f-ing bitter!
• I remain silent.
• I am not in denial.
• Eschew obfuscation.
• When you least expect it, expect it.
• But, I don’t speak English.
• I’m not talking to anyone.
• I can’t talk right now.
• I’m sorry, but I just don’t care.
• Leave me be, I’m asleep.
• I’m speechless.
• I prefer not to have an opinion.
Unquotable

• Know what I hate most? Rhetorical questions. (Camp)

• They all laughed when I said I was going to be a comedian. Well, they’re not laughing now. (Munkhouse)

• If there is anything the nonconformist hates worse than a conformist, it’s another nonconformist who doesn’t conform to the prevailing standard of nonconformity. (Vaughan)

• You know what I hate? Indian givers...no, I take that back. (Phillips)

• Last month I blew $5,000 on a reincarnation seminar. I figured, hey, you only live once. (Shakes)

• I almost had a psychic girlfriend but she left me before we met. Plan to be spontaneous tomorrow. I’d kill for a Nobel Peace Prize. I was trying to daydream, but my mind kept wandering. (Wright)

Movies and TV

• I am at a loss for words. (Ladykillers)

• Me, I’m dishonest! And a dishonest man you can always trust to be dishonest. Honestly, it’s the honest ones you want to look out for. (Pirates of the Carribean)

• I'm not speaking to you! (Will and Grace)

• Crowd: We are all individuals! Man: I’m not. (Life of Brian)
Untitled by Anonymous

There is nothing which cannot be improved

Escher Parody
**Serious Fun**

- There’s a weird idea here: You shouldn’t be able to create a statement that, if interpreted properly, results in another statement that contradicts the original statement.
- It’s a “go back in time and kill your own grandfather” sort of thing.
- There are a bunch of deep mathematical insights that come from applying this idea.
- Here’s a quick survey before the main event.

**Russell’s Paradox**

- In the town of Chelm, there’s a barber. His job is to shave every man in town who does not shave himself.
- Who shaves the barber?
- For the statement to be true:
  - If he shaves himself, then he does not need to shave himself.
  - If he does not shave himself, then he needs to shave himself.
Naive Set Theory

- This example was created to show that there are limitations to how you can create meaningful sets.
- If there is unrestricted self-reference, you can create impossible situations.

Gödel’s Theorem

“This statement cannot be proven.”

- Kurt Gödel showed that any system of mathematics that includes the integers can express this self-referential statement.
  - If it’s true, you can’t prove it! (Incompleteness.)
  - If it’s false, you can prove something that’s false! (Inconsistency.)
- Tough choice.
Kantor’s Diagonalization

- How many fractions are there? *Infinite*.
- How many decimals are there? *Infinite*.
- Are they the same size infinity?
- Well, we can make an infinitely long list that includes every fraction:

List of Fractions

- Start with all the fractions where the numerator and denominator add up to 1, then 2, then 3.

- Every fraction must eventually appear:

<table>
<thead>
<tr>
<th></th>
<th>0/1</th>
<th>1/5</th>
<th>0/9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0/2</td>
<td>2/4</td>
<td>1/8</td>
</tr>
<tr>
<td></td>
<td>1/1</td>
<td>3/3</td>
<td>2/7</td>
</tr>
<tr>
<td></td>
<td>0/3</td>
<td>0/7</td>
<td>3/6</td>
</tr>
<tr>
<td></td>
<td>1/2</td>
<td>1/6</td>
<td>4/5</td>
</tr>
<tr>
<td></td>
<td>0/4</td>
<td>2/5</td>
<td>0/10</td>
</tr>
<tr>
<td></td>
<td>1/3</td>
<td>3/4</td>
<td>1/9</td>
</tr>
<tr>
<td></td>
<td>2/2</td>
<td>0/8</td>
<td>2/8</td>
</tr>
<tr>
<td></td>
<td>0/5</td>
<td>1/7</td>
<td>3/7</td>
</tr>
<tr>
<td></td>
<td>1/4</td>
<td>2/6</td>
<td>4/6</td>
</tr>
<tr>
<td></td>
<td>2/3</td>
<td>3/5</td>
<td>5/5</td>
</tr>
<tr>
<td></td>
<td>0/6</td>
<td>4/4</td>
<td>...</td>
</tr>
</tbody>
</table>
And The Decimals?

- Can we list all the decimals?
- The “add to a constant” trick doesn’t work anymore, since we have decimals like
  0.3333... where the digit sum is infinite.
- So, let’s say we can list them all.
- Here’s the list, hypothetically:

On The List?

- Read down the diagonal.
  0.5898032467...
- Add 1 to each digit (with wraparound).
  0.6909143578...
- The resulting decimal is not on the list!
  (Differs from the $i$th one in the $i$th digit.)
Conclusion

• You can’t make a list of all the decimals.
• You can make a list of all the fractions.
• There are more decimals (real numbers) than fractions (rational numbers)!

Looping Forever

• Given input 10, each of these programs counts backwards from 10 to 1.

• For each, is there an input $n$ we can give that causes the program to loop forever?

```python
def ex1(n):
    for i in range(n, 0, -1):
        print i

def ex2(n):
    while (n > 0):
        print n
        n = n - 1

def ex3(n):
    while (1):
        print n
        n = n - 1
        if (n == 0): return
```
The Halting Problem

• Looping forever is one of the most annoying classes of programming errors.

• Would be great if a tool could automatically detect whether a program always halted.

• We’d like a subroutine `halt` that takes a program as input and returns `true` if the program halts on all inputs and `false` if some input makes it loop forever.

Contrary

• If a `halt` subroutine exists, we can use it to create other programs.

```python
def contrary(prog):
    if prog == contrary:
        if halt(prog) == true:
            while (1):
                print "loop"
        return
```

• For example, this program takes a program “prog” as input, and, if prog is the program `contrary` and `half(prog)` is true, `contrary` loops forever.

• Otherwise it halts.
Contrary Analysis

• What does contrary(contrary) do?

• If halt(contrary) is true, that means contrary halts on all inputs, so it should halt on itself as input.

• But, in this case, contrary(contrary) loops forever!

• So, it must be that halt(contrary) is false, so contrary loops forever on some input.

• But, in this case, note that contrary(prog) halts for any input, including contrary.

Halting Summary

• If contrary(contrary) halts, it loops forever.

• If contrary(contrary) loops forever, it halts.

• As with the Barber paradox, the problem here is our assumption, specifically, that halt exists.

• So, halt is a well-defined problem that no program can solve: It is incomputable.
CS Implications

- There are many problems that turn out to be incomputable.
- All involve computations that might take an infinite number of comparisons to solve and you’re never quite sure when to stop.
- An open problem I posed in my thesis (finding optimal policies for partially observable Markov decision processes) was later shown to be incomputable.

Philosophical Implication

- Some have argued that since people can tell if programs halt but programs can’t tell if programs halt, people are fundamentally more powerful / intelligent than computers.
- Hogwash.
**3x+1 Problem**

- Take a number. Half it if it’s even. Otherwise, triple and add 1. Continue until 1 is reached.

- Any power of 2 will be brought to 1 quickly.

- Some take awhile: 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

- No one knows if it halts on all inputs!

- We can’t (easily) solve the halting problem.

```python
def even(x):
    return not x%2

def collatz(x):
    while (x > 1):
        print x
        if even(x): x = x/2
        else: x = 3*x+1
```

**Subber**

- Here’s an odd little aside.

- For many formal self-reference-based proofs, programs need to be able to refer to themselves.

- How do you do that?

- Consider a subroutine `subber`.

- It takes a string as input and produces a new string as output.

- The output is essentially a copy, but some special characters are converted (subbed).
def subber(q):
    o = ""
    for i in q:
        if i == str(1+1):
            o = o + ""
        elif i == str(1+1+1):
            o = o + q
        elif i == str(1+1+1+1):
            o = o + "\n"
        elif i == str(1+1+1+1+1):
            o = o + "\t"
        elif i == str(1+1+1+1+1+1):
            o = o + "\\n"
        elif i == str(1+1+1+1+1+1+1):
            o = o + "\\\"
        else:
            o = o + i
    return o

def selfPrint():
    print subber("def selfPrint():\nprint subber("def selfPrint():45print subber(232)")")

def selfContained():
    print subber("def subber(q):45o = 2245for i in q:45if i == str(1+1): o = o + '+2'4555elif i == str(1+1+1): o = o + q4555elif i == str(1+1+1+1): o = o + 2624555elif i == str(1+1+1+1+1): o = o + '+2'4555elif i == str(1+1+1+1+1+1): o = o + 282+2i45555elif i == str(1+1+1+1+1+1+1+1): o = o + 28824555else: o = o + i45return o44def selfContained():45print subber(232)")

Self-Referential Program
• Running this program causes it to print precisely the program itself!
• Can even include subber, too:

def selfPrint():
    print subber("def selfPrint():45print subber(232)")

def selfContained():
    print subber("def subber(q):45o = 2245for i in q:45if i == str(1+1): o = o + '+2'4555elif i == str(1+1+1): o = o + q4555elif i == str(1+1+1+1): o = o + 2624555elif i == str(1+1+1+1+1): o = o + '+2'4555elif i == str(1+1+1+1+1+1): o = o + 282+2i45555elif i == str(1+1+1+1+1+1+1+1): o = o + 28824555else: o = o + i45return o44def selfContained():45print subber(232)")

Code
• 2: "
• 3: the whole string
• 4: end of line
• 5: tab
• 6: "\n"
• 7: "\t"
• 8: "\\"
• None of these characters are in subber, by the way.
• Something weird and “birth”-like here. The program has a string, which is the program, which somehow has the string, which is the program...

• Should be infinitely big, but it’s not via clever use of variables and substitution.