CS/MA 109 – Quantitative Reasoning

Wayne Snyder
Computer Science Department
Boston University

Today
Recursion and self-reference: a scientific and culture exploration – and a limitation of algorithms and computers!

Next Tuesday:
Artificial Intelligence Pro: What can computers do, and what will they do in the future?

Next Thursday:
Artificial Intelligence Con: The Turing Test and the limits of computing; The “Singularity” – will it happen??

Final class Tuesday::
Dr. Hall – Review and comments on final exam
Dr. Snyder – Review and comments on final exam
Course Evaluations
Recursion: The Basics

There are basically TWO ways to define something in mathematics (and computer science!):

**Iterative/explicit:** Give an explicit construction, step by step:

Even #s = \{ 0, 2, 4, 6, 8, 10, ...., 2k, ..... \}

N Factorial: \( n! = n \times (n-1) \times ... \times 3 \times 2 \times 1 \)

**Recursive/implicit:** Give a rule for calculating it, based on its structure:

Even #s = “The smallest set containing 0 such that if k is in set, so is (k+2)”

\[
n! = \begin{cases} 
1 & \text{if } n == 1 \\
 n \times (n-1)! & \text{otherwise}
\end{cases}
\]

\[
\begin{align*}
1! & = 1 \\
2! & = 2 \\
3! & = 6 \\
4! & = 24 \\
5! & = 120 \\
\vdots
\end{align*}
\]
Recursion: The Basics

This definition looks suspicious! How can we define something in terms of itself?

“What’s Thai food like?” “Well, it’s like Thai food, only spicier”

“How do I get to Charlestown?” “Well, go up 93, get off at Exit 5, and go towards Charlestown.”

“This statement is false”

There is a long history of amusing visual effects produced by self-reference.....
Recursion: The Basics
Recursion: The Basics
Recursion: The Basics
But actually, these are examples of “infinite recursion” where the self-reference creates an infinite digression. This is equivalent to a loop that does not stop.

A good recursive definition of an entity X looks like this:

**Base case:** Define simple cases of X explicitly, without reference to itself;

**Recursive case:** Define X in terms of simpler versions of itself, which are closer to the base case.

The key element here is that the recursion must stop at the base case, just as an iterative loop must stop at some point.
Recursion: Implementation

Let’s look at an example, the Fibonacci numbers, a sequence easily defined as “write down two 1s; for each succeeding number, write down the sum of the previous two”:

\[
F_0 \quad F_1 \quad F_2 \quad F_3 \quad F_4 \quad F_5 \quad F_6 \quad F_7 \quad F_k = F_{k-1} + F_{k-2}
\]

1, 1, 2, 3, 5, 8, 13, 21, .... (sum of previous 2 terms)...

Digression: The Fibonacci Numbers have a long history; the earliest mention is in an analysis of Sanskrit poetry, c. 200 AD, but the name comes from Leonardo of Pisa, better known as Fibonacci, who invented them to explain the geometric growth of a family of rabbits. We will return to some of the deep properties of this sequence in a moment......
Recursion: Implementation

Let’s look an example, the Fibonacci numbers, a sequence easily defined as “write down two 1s; for each succeeding number, write down the sum of the previous two”:

1, 1, 2, 3, 5, 8, 13, 21, .... (sum of previous 2 terms)...

\[ F_0 \quad F_1 \quad F_2 \quad F_3 \quad F_4 \quad F_5 \quad F_6 \quad F_7 \quad F_k = F_{k-1} + F_{k-2} \]

// Iterative definition of Kth Fibonacci number

\[ \text{Fibonacci}(K) : \]

1. Set F equal to 1 and PrevF equal to 1;
2. If K = 0, stop and output F;
3. Set Temp equal to F + PrevF;
4. Set PrevF equal to F and set F equal to Temp;
5. Subtract 1 from K and go to (2).

\[ F_5 = \text{Fibonacci}(5) => 6 \]
Recursion: Implementation

Fibonacci Numbers:

1, 1, 2, 3, 5,....

\[ F_0 \quad F_1 \quad F_2 \quad F_3 \quad F_4 \quad F_k = F_{k-1} + F_{k-2} \]

Recursive Definition of \( F_k \):

\[
F_k = \begin{cases} 
1 & \text{if } k < 2 \\
F_{k-1} + F_{k-2} & \text{otherwise}
\end{cases}
\]

// Recursive algorithm

```c
fibonacci( K )
1. If K = 0 or K = 1, stop and output 1;
2. Let T = fibonacci(K-1) + fibonacci(K-2);
3. Stop and output T.
```

\( F_5 = \text{fibonacci}(5) \Rightarrow 6 \)
Recursion and Self-Reference

Recursion is an example of self-reference in algorithms.....

Recursive Definition of $F_k$:

$F_k = \begin{cases} 1 & \text{if } k < 2 \\ F_{k-1} + F_{k-2} & \text{otherwise} \end{cases}$

// Recursive algorithm

fibonacci( K ) {
    1. If K = 0 or K = 1, stop and output 1;
    2. Let T = fibonacci(K-1) + fibonacci(K-2);
    3. Stop and output T.
Self-Reference in the Arts: The Golden Ratio

In fact, recursion and self-reference, in the form of the Golden Ratio, has had a long history in art, music, and architecture….

The Golden Ratio is an irrational number 1.61803399….. which is the limit of the ratios of successive Fibonacci Numbers, and represents the “perfect ratio” of two quantities a and b:

\[
\frac{a}{b} = \frac{(a+b)}{a} = 1.618…
\]

\[
\text{a+b is to a as a is to b}
\]

<table>
<thead>
<tr>
<th>K</th>
<th>F&lt;sub&gt;K&lt;/sub&gt;</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1.67</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>1.6</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>1.625</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>1.61538462</td>
</tr>
<tr>
<td>7</td>
<td>34</td>
<td>1.61904762</td>
</tr>
<tr>
<td>8</td>
<td>55</td>
<td>1.61764706</td>
</tr>
<tr>
<td>9</td>
<td>89</td>
<td>1.61818182</td>
</tr>
<tr>
<td>10</td>
<td>144</td>
<td>1.61797753</td>
</tr>
<tr>
<td>11</td>
<td>233</td>
<td>1.61805556</td>
</tr>
<tr>
<td>12</td>
<td>377</td>
<td>1.61802575</td>
</tr>
<tr>
<td>13</td>
<td>610</td>
<td>1.61803714</td>
</tr>
<tr>
<td>14</td>
<td>987</td>
<td>1.61802379</td>
</tr>
<tr>
<td>15</td>
<td>1597</td>
<td>1.61803445</td>
</tr>
<tr>
<td>16</td>
<td>2584</td>
<td>1.61803381</td>
</tr>
<tr>
<td>17</td>
<td>4181</td>
<td>1.61803406</td>
</tr>
<tr>
<td>18</td>
<td>6765</td>
<td>1.61804396</td>
</tr>
<tr>
<td>19</td>
<td>10946</td>
<td>1.61803934</td>
</tr>
<tr>
<td>20</td>
<td>17711</td>
<td>1.61803399</td>
</tr>
<tr>
<td>21</td>
<td>28657</td>
<td>1.61803399</td>
</tr>
<tr>
<td>22</td>
<td>46368</td>
<td>1.61803399</td>
</tr>
<tr>
<td>23</td>
<td>75025</td>
<td>1.61803399</td>
</tr>
</tbody>
</table>
Recursion: Implementation

\[ a + b \]

where \( a + b \) is to \( a \) as \( a \) is to \( b \).

Golden Ratio: \( 1.61803 \ldots \)
Recursion: Implementation

Golden Ratio: 1.6180339....

\[ a + b \text{ is to } a \text{ as } a \text{ is to } b \]
Self-Reference in the Arts: The Golden Ratio

Hm… maybe…. what do you think, which of the following rectangles has the most pleasing proportions?
Self-Reference in the Arts: The Golden Ratio

Hm… maybe…. what do you think, which of the following rectangles has the most pleasing proportions?

- 1 : 1.618
- 2 : 3
- 1 : 2
Computer Screens after 2000 tended towards a screen size of 1.6 : 1, although after 2010 they moved towards 16:9 or 1.78:1.

In fact, the usable screen area of my macbook pro is 332mm x 204mm, which is 1.627 : 1.
The Golden Ratio was “discovered” to be the basis for the proportions of the beautiful human body.....
Self-Reference in the Arts: The Golden Ratio

..... for many repetitive patterns in nature.....
Self-Reference in the Arts: The Golden Ratio

..... And also in things that are unnatural!
Self-Reference in the Arts: The Golden Ratio

It was “discovered” to exist in many works of art, music, and architecture....
Self-Reference in the Arts: The Golden Ratio

In any case, the Golden Ratio has been used *consciously* by artists and writers throughout history…
Quasi-crystals and the Golden Ratio

Quasi-crystals represent a newly discovered state of matter.

Most crystals in nature, such as those in sugar, salt or diamonds, are symmetrical and all have the same orientation throughout the entire crystal. Quasicrystals represent a new state of matter that was not expected to be found, with some properties of crystals and others of non-crystalline matter, such as glass.
Self-reference in language has already been mentioned, through a statement called the Liar’s Paradox:

“This statement is false”

This one statement has appeared throughout history to illustrate of the problem of a language that can reflect on itself:

“All Cretans are liars” -- Epimenides the Cretan [The “Epimenides Paradox”]

New Testament (Titus 1:12-13): ”One of Crete's own prophets has said it: ‘Cretans are always liars ….. ’ ”

“The town barber is the man who shaves all, and only, those men who don’t shave themselves. Who shaves the town barber?” [The “Barber Paradox”]

Apparently this can also be used as a weapon to defeat those who are excessively logical..... http://www.youtube.com/watch?v=EzVxsYzXI_Y
Self-reference can be used for humor (or attempts thereof) as well as serious philosophy:

**Autological Words** are those which describe themselves:

“Unhyphenated” has no hyphens;  
“Pentasyllabic” has five syllables;  
“Sesquipendalian” is a long word;  
“Mispeled” is an autological word which is also an example of an error:

**Fumblerules** are grammatical rules which self-referentially violate themselves:

“Avoid cliches like the plague.”

“The passive voice should not be used.”

“Prepositions are not words to end a sentence with.”

**Hofstadter’s Law**: “It always takes longer than you think, even when you take Hofstadter’s Law into account.”
Self-Reference in the Fine Arts

One of the major characteristics of 20th Century arts and literature is its reflection on the process of artistic expression....
There is even a genre of literature called Metafiction, in which the authors self-referentially refers to the story or novel or play:

A story about a writer who is writing a story;

A story in which the characters are aware that they are in a story;

A play in which the audience plays a role in choosing how the play will end;

Examples (among many):

Pirandello’s *Six Characters in Search of an Author*;

Kurt Vonnegut’s *Slaughterhouse-Five*: “All this happened, more or less ... that was I, that was me. That was the author of this book....”

Many, many movies!
Self-reference has had a profound effect on the development of modern mathematics and computer science, starting with a formulation of the Liar’s Paradox in the mathematical theory of sets:

Let \( R = \{ x \mid x \notin x \} \), then \( R \in R \iff R \notin R \) [The “Russell Paradox”]

The notation for sets \( \{ x \mid P(x) \} \) is a kind of description: “The set of all \( x \) which satisfy the statement \( P(x) \).” There are thus two kinds of set descriptions:

**Autologous:** Those which contain themselves (“Sentences containing four words”)
**Non-autologous:** Those which do not (“Sentences containing five words”)

### All Set Descriptions:

<table>
<thead>
<tr>
<th>Autologous:</th>
<th>Non-autologous:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentences containing four words.</td>
<td>Sentences containing five words.</td>
</tr>
<tr>
<td>Sentences beginning with the letter ‘S’.</td>
<td>Verbs. ( { x \mid x \text{ has exactly 4 members} } )</td>
</tr>
<tr>
<td>Heptasyllabic phrases.</td>
<td></td>
</tr>
<tr>
<td>( { x \mid x \text{ has more than 1 member} } )</td>
<td></td>
</tr>
</tbody>
</table>
The Russell Paradox:

There are two kinds of set descriptions:

**Autologous:** Those which contain themselves ("Sentences containing four words")
**Non-Autologous:** Those which do not ("Sentences containing five words")

**Question:** Which category does the collection of Non-Autologous Descriptions belong to, i.e., the collection of all collections which do not contain themselves?

Bertrand Russell was considering the power of descriptions, and showed that a sufficiently complicated language for descriptions would inevitably fail to describe some things.

Human languages, music, literature, and movies are all such sufficiently complicated languages...

But so is mathematics and computer science.....
Gödel’s Incompleteness Theorem

Kurt Gödel, at age 25, proved that any sufficiently complex mathematical system will be consistent with true mathematical statements which can NOT BE PROVED within the system.

Such a mathematical system has to have only basic operators such as +, *, -, /, “for all”, “implies”, e.g.,

“All numbers are either prime or non-prime”

“Every number divisible by 4 is divisible by 2.”

Such sentences can be proved from a set of axioms.

Some mathematical statements are true but can never be proved by mathematics!
Turing’s Halting Problem shows that computers are subject to the same limitations.…

There is no absolutely certain way of determining, for an arbitrary algorithm, whether it ever halts and returns an answer.
Self-Reference in Mathematics

**Proof by Contradiction:** Suppose we had an algorithm A which could decide, for any OTHER algorithm B and input L, whether B halts on L:

![Diagram of algorithm A with inputs and outputs]

- **Algorithm B**
- **Input L**
  - \( B \): \( \sqrt{2} \)
  - \( L: \ 2 \)
  - \( L: \ 1.4142... \)
  - \( L: \ 2 \)
  - \( B': \) **Loops Forever!**
  - \( \) **??**

- **Google Halting Checker**
  - **A**
  - **Yes, it halts!**
  - **No, it doesn’t halt!**
Suppose we had an algorithm A which could decide, for any OTHER algorithm B and input L, whether B halts on L:

**B**

√

**L**

2

A

Google Halting Checker

Yes, it halts!

No, it doesn’t halt!
Suppose we had an algorithm A which could decide, for any OTHER algorithm B and input L, whether B halts on L:

- **B'**
  - Loops Forever!

- **L**
  - 2

- **Google Halting Checker**

  - Yes, it halts!
  - No, it doesn’t halt!
Then we make a small change to A: instead of returning the answer “Yes, it halts,” when the input algorithm halts, A runs B’ (which doesn’t halt). Let us call this A’:

This does the opposite of B! If B halts, then A does not; if B does not halt, then A halts and says “No, it doesn’t halt!”
Now to send this down the black hole of recursion: we .... wait for it...... give A’ a copy of itself and ask it to check whether itself halts:

If A’ halts on itself, then it runs B’ and doesn’t halt!

If A’ doesn’t halt, then it halts and says “No, it doesn’t halt!”

Contradition: So no such program A’ can ever exist!
Consequences of the Halting Problem:

Not all decision problems ("yes/no problems") can be solved; some problems are undecidable.

We may be able to prove that some computer programs work as expected, but there is no absolutely certain method for determining whether an arbitrary program works properly.

For sufficiently complicated programs, we may design them carefully and test them thoroughly, but we will never know for absolute certain that they work properly……

All Problems

Unsolvable: it is a logical contradiction that such problems can ever be solved!

Solvable: A algorithm exists which will terminate with the correct answer.

Practically Solvable: An algorithm exists which can solve the problem in a polynomial number of steps.