Lecture 2.8: Russell's paradox and the haulting problem

Matthew Macauley

Department of Mathematical Sciences
Clemson University
http://www.math.clemson.edu/~macaule/

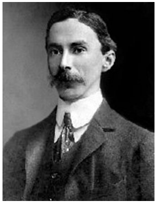
Math 4190, Discrete Mathematical Structures

Russell's paradox

Recall Russell's paradox:

Suppose a town's barber shaves every man who doesn't shave himself.

Who shaves the barber?



This was due to British philsopher, logician, and mathematician Bertrand Russell (1872–1970), as an informal way to describe the paradox behind the set of all sets that don't contain themselves:

$$S = \{A \mid A \text{ is a set, and } A \notin A\}.$$

If such a set exists, then

$$S \in S \longrightarrow S \notin S, \qquad S \notin S \to S \in S,$$

which is a contradiction.

A "fix" for Russell's paradox

Let U be a universal set, and let

$$S = \{A \mid A \subseteq U \text{ and } A \not\in A\}.$$

Now, suppose that $S \in S$. Then by defintion, $S \subseteq U$ *and* $S \not\in S$.

If $S \not\in S$, then

$$\neg(S \subseteq U \quad \wedge \quad S \not\in S) \quad \Longrightarrow \quad (S \not\subseteq U \quad \vee \quad S \in S) \quad \Longrightarrow \quad S \not\subseteq U.$$

In other words, there cannot exist a "set of all sets", but we can define a "class of all sets".

One may ask whether we can always get rid of contradictions in this manner. In 1931, Kurt Gödel said "no."

Gödel's incompleteness theorem (informally)

Every axiomatic system that is sufficient to develop the basic laws of arithmetic cannot be both consistent and complete.

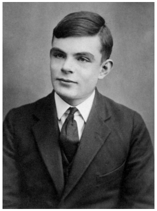
In other words, there will always be either paradoxes, or statements that are unprovable.

The halting problem

Alan Turing (1912–54) was one of the pioneers of the field of computer science.

During World War II, he decripted German intelligence codes for the British government, which played a major role in the Allies defeating the Nazis.

One of Turing's many contributions to computer science is the formalization of the algorithm.



Theorem(Turing, 1930s)

There *cannot exist* a computer algorithm that accepts any algorithm X, and data set D, and outputs:

- "HALTS", if X halts with input D,
- "LOOPS FOREVER", if X loops forever with input D.

Turing proved this by contradiction. He supposed there existed such an algorithm, and showed that it lead to an instance of Russell's paradox.

The halting problem

Theorem (Turing, 1930s)

There *cannot exist* a computer algorithm that accepts any algorithm X, and data set D, and outputs:

- "HALTS", if X halts with input D,
- "LOOPS FOREVER", if X loops forever with input D.

Proof

Suppose there exists such an algorithm:

```
CheckHalt(X, D) {
    if (X terminates in finitely many steps given D)
        print HALTS;
    else
        print LOOPS FOREVER;
    end if
}
```

Note that we could run **CheckHalt**(X, X).

The halting problem

Theorem (Turing, 1930s)

There *cannot exist* a computer algorithm that accepts any algorithm X, and data set D, and outputs:

- "HALTS", if X halts with input D,
- "LOOPS FOREVER", if X loops forever with input D.

Proof (contin.)

Define the following algorithm:

```
Russell(X) {
    if (CheckHalt(X,X) prints "HALTS")
        while 1;     \\ loops forever
    else if (CheckHalt(X,X) prints "HALTS")
        halt;
    end if
}
```

Now, consider **Russell**(**Russell**).

The halting problem

Theorem (Turing, 1930s)

There *cannot exist* a computer algorithm that accepts any algorithm X, and data set D, and outputs:

- "HALTS", if X halts with input D,
- "LOOPS FOREVER", if X loops forever with input D.

Proof (contin.)

What happens when we run **Russell**(**Russell**)?

Case 1. **Russell**(**Russell**) terminates after finitely many steps.

Then **CheckHalt**(Russell, Russell) prints "HALTS".

This means that **Russell**(**Russell**) loops forever.

Case 2. **Russell**(**Russell**) loops forever.

Then **CheckHalt**(Russell, Russell) prints "LOOPS FOREVER".

This means that **Russell**(**Russell**) halts.

This is a contradiction, and it means that the **CheckHalt** algorithm cannot exist. $\square$

Undecidable problems

The halting problem is said to be undecidable.

It requires a yes/no answer, but there cannot exist any algorithm to answer it.

There are many other problems that have been shown to be undecidable.

- *Abstract algebra*: Determine whether two groups are isomorphic.
- *Topology*: Determine whether two 4-dimensional topological spaces are homeomorphic.
- *Number theory*: Determine whether a Diophantine equation has a solution over $\mathbb{Z}$.
- *Linear algebra*: Given a finite set of $n \times n$ matrices over $\mathbb{Z}$, can they be multiplied to yield the zero matrix?
- *Computational linguistics*: Determine whether two context-free grammars generate the same set of strings.
- *Computer science*: Determine the Kolmogorov complexity of a string.

Additionally, many classic unsolved problems may very well be unprovable, such as P vs. NP, the Riemann hypothesis, the $3n + 1$ problem, and the Goldbach conjecture.