Welcome to CS 39
Theory of Computation

Professor Amit Chakrabarti
Teaching Assistant: Vibhor Bhatt
http://www.cs.dartmouth.edu/~cs39

Russell’s Paradox
• Mr. Jones in the only barber in town. He shaves all those men and only those men who do not shave themselves.
• If YES, Mr. Jones shaves himself…
  – by his own condition, he cannot shave himself!
• If NO, Mr. Jones does not shave himself…
  – by his own condition, he must shave himself!!

Math notation: Logic
Variables $x, y, \ldots$ stand for polygons in the plane.
• Let $p(x)$ be the statement “$x$ is a parallelogram”.
• Let $q(x) = “x$ has at least one right angle”.
• Let $r(x) = “x$ is a rectangle”.
• Let $s(x) = “x$ is a square”.
• Let $t(x) = “x$ is a rhombus”.

True or false: $p(x) \land q(x) \Rightarrow r(x)$

True
Let $p(x)$ be the statement “$x$ is a parallelogram”.
Let $q(x) =$ “$x$ has at least one right angle”.
Let $r(x) =$ “$x$ is a rectangle”.
Let $s(x) =$ “$x$ is a square”.
Let $t(x) =$ “$x$ is a rhombus”.

True or false: $r(x) \land s(x) \Rightarrow t(x)$

In fact $s(x) \Rightarrow t(x)$

True or false: $t(x) \Rightarrow \neg r(x)$

Not all rhombuses are rectangles, but some are. $t(x) \not\Rightarrow r(x)$

Math notation: Quantifiers

- Strictly speaking, we should use quantifiers whenever there seems to be ambiguity.
- $\forall:$ “for all” (universal quantifier)
- $\exists:$ “there exists” (existential quantifier)

Thus, $\forall x (t(x) \Rightarrow \neg r(x))$ is false.
  - Because not all rhombuses are non-rectangles.
But $\exists x (t(x) \Rightarrow \neg r(x))$ is true.
  - Because there do exist rhombuses that are non- rects.

Negating an implication gives an AND statement
$\neg(P \Rightarrow Q)$ is the same as $(P \land \neg Q)$, which is the same as $(P \land \neg Q)$.

Negating a quantified statement flips the quantifier type
$\neg \forall x (t(x) \Rightarrow \neg r(x))$ is the same as $\exists x (\neg (t(x) \Rightarrow \neg r(x)))$,
which is the same as $\exists x (t(x) \land r(x))$.

Does this make sense? If not, say so now!
Types of proof

- Proof by construction
- Proof by contradiction
- Proof by (mathematical) induction

- Which type of proof is best?
  - This question makes no sense.
  - Any proof style is good, so long as you write complete and rigorous proofs.
  - In fact, within a single long proof you may want to use two, or all three, styles.

Proof by construction

Theorem: Every even positive integer can be written as the sum of two odd positive integers.

Proof: Let $2m$ be an even positive integer.
If $m$ is odd, write $2m = m + m$, and we are done.
If $m$ is even, write $2m = (m–1) + (m+1)$ …
… since $m$ is even, $m \geq 2$, so $(m–1)$ is positive … and we are done.

Proof by induction

To prove a theorem by induction:
- Prove the theorem for a general case by assuming the same theorem to be true (“induction hypothesis”) for all smaller cases.
- Separately prove the theorem, without making any assumptions, for all “base” cases, i.e., those cases for which there is nothing smaller.
  - Many people prefer to write the base case(s) first.

Proof by induction

Theorem: Every even positive integer can be written as the sum of two odd positive integers.

Base case: 2 can be written as $1+1$.
Induction step: Let $m$ be an even integer $> 2$.
… Then $(m–2)$ is a smaller even positive integer.
… By induction hypothesis, $m–2 = p+q$ for odd $p, q > 0$.
… So, $m = (p+2) + q$, i.e., $m$ can also be written as the sum of two odd positive integers.
Proof by contradiction

• We want to prove a statement \( S \).

• Begin by assuming the opposite of \( S \) (i.e., \( \neg S \)).

• Use logical reasoning to arrive at a contradiction.

• We are then forced to conclude that our assumption is wrong, i.e., that \( S \) is true.

Infinite Loop Tester

• You are a grader for CS 5

• Students submit a program “foo.java”
  – You have test input files “1.inp”… “100.inp”
  – Every test case correctly handled: 1 point
  – Infinite loop: -20 points (penalty)

• You wish to automate the grading process
  – Write a program “ILT.java” which takes two input files ---“foo.java” and “i.inp” --- and checks whether “foo.java” enters an infinite loop on input “i.inp”.

Proof by contradiction

• Let \( P(I) \) denote program \( P \) running on input \( I \).

• We’ll assume that ILT can be written.

• Recall that \( ILT(P,I) \) says “infinite loop” if \( P(I) \) enters an infinite loop. Otherwise it says “halts”.

• For our logical reasoning, we’ll construct a program Buster, using the program ILT.

Buster(P) does the following:

1. Run \( ILT(P,P) \).
2. If \( ILT \) says “infinite loop”, then halt.
3. If \( ILT \) says “halts”, enter an infinite loop!

What happens when we run Buster(Buster)?

– If the run halts, then in line 1 ILT would have said “halts”, so we would go to line 3 and…OOPS!
– If the run does not halt, then after line 1 we’d go to line 2 and… OOPS!

We have a contradiction. Thus, \( ILT \) cannot exist.
Think it over

• The proof you have just seen is one of the most profound results in the theory of computing.

• Make sure you understand it.

• Try to explain the proof to a friend.