Boolean Logic

Pete Manolios
Northeastern

Logic and Computation, 1/30/2019

Objectives

- Q&A
- P vs NP
- Properties of Boolean Operators

P vs NP

- Is there an algorithm that given a Boolean formula returns “yes” if it is SAT else “no”?
- Easy. Construct truth table
- But the complexity is exponential
  - The number of rows in the truth table is 2^n, where n is the number of atoms
- Is there an efficient algorithm for determining Boolean satisfiability?
  - Efficient means an algorithm that in the worst case runs in polynomial time
  - Godel asked this question in a letter he wrote to von Neumann in 1956
  - No one knows, although this is one of the most studied questions in CS
  - Most experts believe that no polynomial time algorithm exists
- An assignment is a certificate for SAT: using it to check satisfiability is “easy” (P-Time)
  - But, coming up with a satisfying assignment is “hard”
  - If we checking solutions to a problem is easy, is it also easy to solve the problem?
- There is a large class of “hard” problems that can be solved efficiently if SAT can be solved efficiently

Clay Institute Millennium Problems

CMI

ABOUT PROGRAMS MILLENNIUM PROBLEMS PEOPLE PUBLICATIONS EVENTS EUCLID

Purpose and Goals

The Clay Mathematics Institute is a tax-exempt Private Operating Foundation dedicated to increasing and disseminating mathematical knowledge. It supports the work of leading researchers at various stages of their careers and organizes conferences, workshops, and summer schools. Contemporary breakthroughs are recognized by its annual Research Award.

The Clay Mathematics Institute (CMI) has named seven "Millennium Prize Problems." The Scientific Advisory Board of CMI (SAB) selected these problems, focusing on important classic questions that have resisted solution over the years. The Board of Directors of CMI designated a $7 million prize fund for the solutions to these problems, with $1 million allocated to each. The Directors of CMI, and no other persons or body, have

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Solution

- Which is the simplest expression equivalent to $\neg(p \oplus p \Rightarrow q)$?

```
   ¬(p ⊕ (p ⇒ q))
≡  { ¬(p⊕q) ≡ (p≡q)}
   (p ≡ (p ⇒ q))
≡  { Shannon }
   (p ∧ (true ≡ (true ⇒ q))) ∨ (¬p ∧ (false ≡ (false ⇒ q)))
≡  { Constant Prop }
   (p ∧ q) ∨ (¬p ∧ false)
≡  { Constant Prop }
   p ∧ q
```

Properties

- Review lecture notes

Equational Proofs

- We are going to use equational proofs throughout the semester!
- An equational proof is just a sequence of equality preserving transformations
- To show that `f=g` is valid, we have a proof of the form:

```
   f
=  { hint 1 }
   f1
=  { hint 2 }
   . . .
=  { hint n+1 }
   g
```

- If `g` is a validity (e.g., `true`), then this is a proof that `f` is valid
- Hints should contain enough information to understand the equality

Equational Proofs

- If the formulas are Boolean, we can instead use this form

```
   f
≡  { hint 1 }
   f₁
≡  { hint 2 }
   . . .
≡  { hint n+1 }
   g
```

- By transitivity of = and ≡
  - if f = (≡) f_1 and … and f_n = (≡) g, then
  - then f = (≡) g

Equational Proofs

- If we have a transitive operator, say ⇒, then this also works

```
   f
⇒   { hint 1 }
   f₁
⇒   { hint 2 }
   . . .
⇒   { hint n+1 }
   g
```

- By transitivity of ⇒ and ≡
  - if $f \Rightarrow f_1$ and … and $f_n \Rightarrow g$, then $f \Rightarrow g$
- Other transitive operators include <, >, ≤, ≥, etc.

Your Mission ...

- Review lecture notes
- You have to memorize the rules
  - Repeat
    - Read a section
    - Close lecture notes and write down equalities
    - Until: you can write equalities without thinking
  - Similar to learning your multiplication tables
- Instantiation is used implicitly all the time
- Associativity/commutativity is used all the time
- While truth tables can be used for almost everything, they are too slow, so you have to get comfortable with equational reasoning