Boolean Logic

Part 6

1

Logic Statements

Make Mr. Spock Proud

2

Logic Statements

- Logic is used to construct all proofs and computer systems
- A *statement* is any declarative sentence that results in either true or false

3

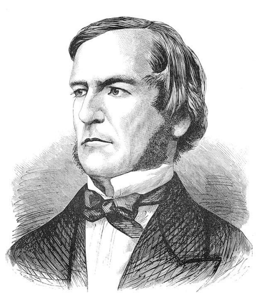
Examples of Statements

- *There are exactly 35 people in this room*
- *Sacramento State is located next to a river*
- *10 + 2 = 11*
- *We have great choices for the 2020 Election*

4

Boolean Logic

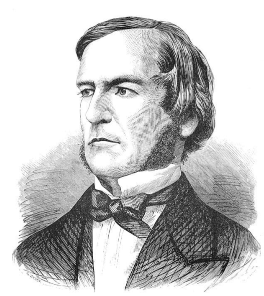
- Discovered by George Boole
- First published in *The Mathematical Analysis of Logic (1847)*



5

Boolean Logic

- Revolutionized logic & proofs and is part of framework of modern of computer science



6

Boolean Operators

- Statements can be combined in *compound statements* using Boolean operators
- After statements are combined, they are still statements
- For example: "p and q"
  - given that p and q are both statements
  - then "p and q" is also a statement

7

Let's Review Boolean Operators

Operator	Name
`p and q`	True only if both p and q are true
`p or q`	True if either p or q true
`not p`	True if p false
`p xor q`	True if p and q are different
`p implies q`	True unless p is true and q is false

8

Logic Notation of Operators

Logic	C Family	Visual Basic
$p \land q$	`p && q`	`p and q`
$p \lor q$	`p || q`	`p or q`
$\neg p$	`!p`	`not p`
$p \oplus q$	*none*	`p xor q`

9

Truth Tables

- Truth tables are useful tools for analyzing a large Boolean expression
- The table includes all the possible combinations of True and False for each input into the equation
- This results in 2^n rows where n is the number of inputs

10

Truth Table

p	q	$\neg p$	$p \lor q$	$p \land q$	$p \to q$
T	T	F	T	T	T
T	F	F	T	F	F
F	T	T	T	F	T
F	F	T	F	F	T

11

AND Example

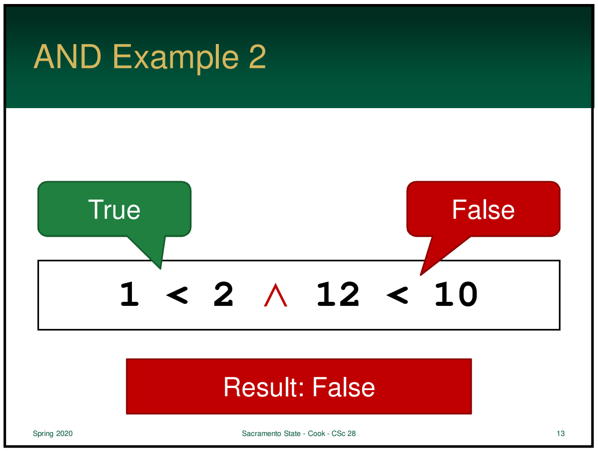
True

True

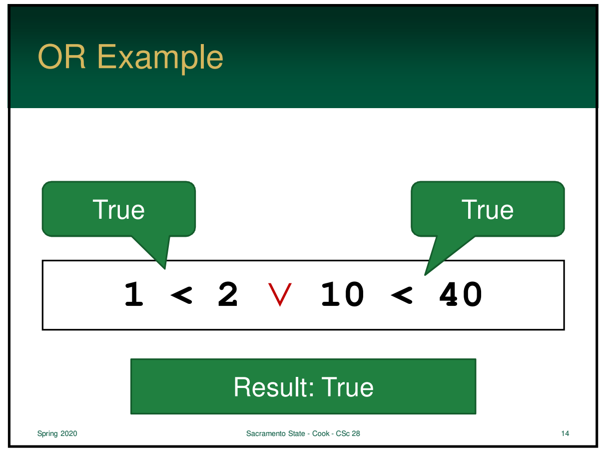
$$1 < 2 \land 10 < 40$$

Result: True

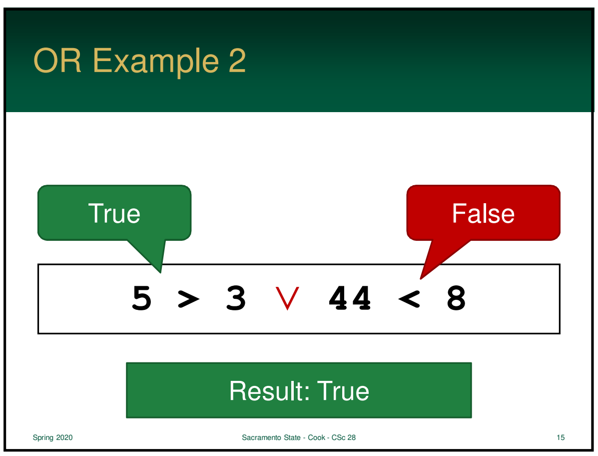
12



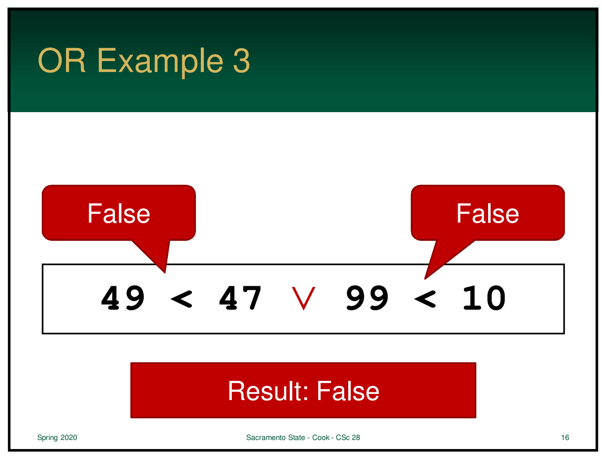
13



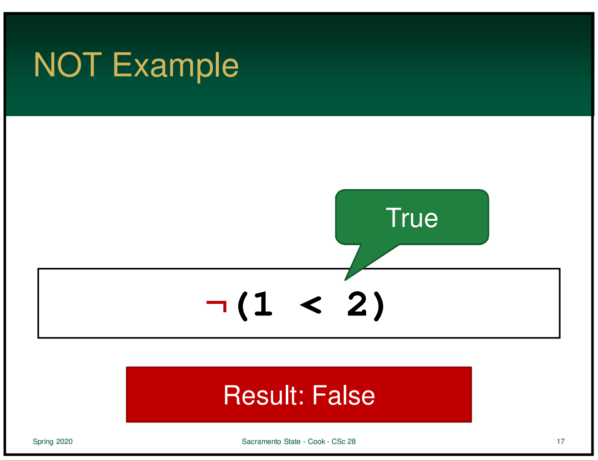
14



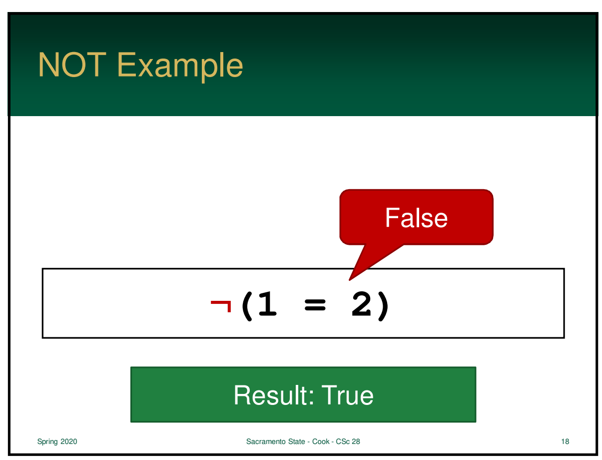
15



16



17



18

Examples

$1 < 3 \land 10 < 40$	True
$1 = 3 \land 10 < 40$	False
$\neg (12 \neq 12)$	True
$1 > 3 \lor 30 < 20$	False

19

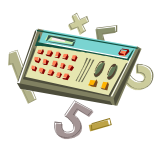
Logical Operator Precedence

Yup, we have that here too!

20

Logical Operator Precedence

- In *propositional logic*, statements can be combined with other statements using logical operators
- So, they can be chained together to form complex logic



21

Algebra: Order of Operations

- Some mathematical operators have a high "precedence" than others
- They are computed first

```
3 + 6 / 3 * 2
```


22

Algebra: Order of Operations

- Knowing the correct order is vital
- For example, what is the result of the expression below?

```
3 + 6 / 3 * 2
```


23

Algebra: Order of Operations

- It is 7
- Divide and multiply are <u>equal</u> (and then done left to right), addition is done last

```
3 + 6 / 3 * 2  =  7
```


24

Many try, many fail

- Many students, using "PEMDAS" , think multiply is done before divide (M is before D)
- … or they just go left to right with no regard to precedence

```
3 + 6 / 3 * 2  =  4    WRONG! F-!
```


25

Standard Precedence Levels

1	¬
2	∧
3	∨ ⊕

Highest Level

Lowest Level

26

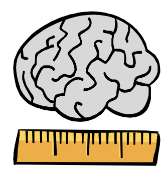
Defining Boolean Logic

"Want to define it?"
"True."

27

Defining Boolean Logic

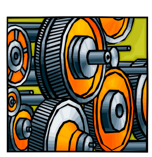
- Let's look at what exactly Boolean logic is in context of data types and functions
- Once we define the Boolean Data type, we can apply it to other systems



28

Functions

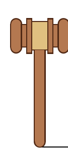
- Also recall functions from earlier
- An *abstract data type* is a set of values and functions on those values
- So, we can define the data type for Boolean values



29

Extending Boolean to Other Types

- The abstract data type for a Boolean Data Type can be written as a 6-tuple



$$B = (S, \vee, \wedge, \neg, \bot, \top)$$

30

Extending Boolean to Other Types

- The first property is a set S contains two elements
- These are: $\perp$ (smaller) and $\top$ (bigger)

$$B = (S, \vee, \wedge, \neg, \perp, \top)$$

31

Extending Boolean to Other Types

- 3 operations on the set S: $\vee$, $\wedge$, $\neg$
- Must follow the 4 primary axioms: Identity, Communitive, Distributive, Complement

$$B = (S, \vee, \wedge, \neg, \perp, \top)$$

32

Defining Boolean Logic

- Boolean Logic is closely related to Set Theory
- So much, in fact, that Boolean Logic can be considered as a special case of sets

{ }

33

Defining Boolean Logic

- We can show that the behavior of Boolean Logic can be created in sets
- It's not surprised that many of the laws for sets work for Boolean Logic

{ }

34

Set Theory & Boolean Logic

- First, we can define True as the Universe
- Remember that in binary logic, it simply *is* or it *isn't*
- So, the Universe means 1 or true

$\top = \mathbf{U}$

$F = \varnothing$

35

Set Theory & Boolean Logic

- The complement of $\mathbf{U}$ is $\varnothing$
- So, naturally, False is represented with $\varnothing$

$\top = \mathbf{U}$

$F = \varnothing$

36

Boolean Operators with Sets

- The Union operator is analogous to the And operator
- Likewise, Intersection is analogous to Or.

$a \wedge b = A \cap B$

$a \vee b = A \cup B$

37

And-Intersection Comparison

Boolean Logic	Set Theory
$T \wedge T = T$	$U \cap U = U$
$T \wedge F = F$	$U \cap \varnothing = \varnothing$
$F \wedge T = F$	$\varnothing \cap U = \varnothing$
$F \wedge F = F$	$\varnothing \cap \varnothing = \varnothing$

38

Or-Intersection Comparison

Boolean Logic	Set Theory
$T \vee T = T$	$U \cup U = U$
$T \vee F = T$	$U \cup \varnothing = U$
$F \vee T = T$	$\varnothing \cup U = U$
$F \vee F = F$	$\varnothing \cup \varnothing = \varnothing$

39

Boolean Not with Complement

- The Boolean Not operator can also be implemented using set theory
- In this case, complement

$\neg a = A'$

40

Or-Intersection Comparison

Boolean Logic	Set Theory
$\neg T = F$	$U' = \varnothing$
$\neg F = T$	$\varnothing' = U$

41

The Axioms

- The Axioms are:
  - Identity
  - Communitive
  - Distributive
  - Complement
- The axioms from Set Theory apply to Boolean Logic

42

Tautology & Contradiction

When the logic is quite elementary

43

Tautology & Contradictions

- Some statements are always true or false regardless of the variables used
- If the statement is always true, it is called *tautology*
- If the statement is always false, it is called a *contradiction*

44

Example Tautologies

- The following are examples of tautologies
- The result will always be **true**

```
p ∨ ¬p
p → p
p = p
```


45

Example Contradictions

- The following are examples of contradictions
- The result will always be **false**

```
p ∧ ¬p
p ⊕ p
```


46

Example

- So, what is the truth table for the example below?
- Let's create a truth table

```
(¬p ∧ q) ∧ (p ∨ ¬q)
```


47

Logic Equivalence

Same meaning, different form

48

Logical Equivalence

- *Logical equivalence* is when two different statements are the same
- The truth tables for both statements are identical

49

First Four Axioms

- The first four fundamental axioms were developed by *Edward Huntington* in 1904
- Other rules are derived from these

50

Communitive Law

- Both $\land$ and $\lor$ are communitive
- This means the left-hand and right-hand operands can be switched (symmetric relation)

$$a \land b \equiv b \land a$$
$$a \lor b \equiv b \lor a$$

51

Identity Law

- Some operands will have no effect on the truth table of a statement
- In this case, the statement can be simplified

$$a \land \text{true} \equiv a$$
$$a \lor \text{false} \equiv a$$

52

Complement Law

- When a statement is used with its complement (itself negated), it will result in either true or false
- So, it is always as tautology or contradiction

$$a \land \neg a \equiv \text{false}$$
$$a \lor \neg a \equiv \text{true}$$

53

Distributive Law

- Math has operators that are *distributive*
- For example: $a * (b + c) = (a * b) + (a * c)$
- Works for both $\land$ and $\lor$

$$a \land (b \lor c) \equiv (a \land b) \lor (a \land c)$$
$$a \lor (b \land c) \equiv (a \lor b) \land (a \lor c)$$

54

Logical Equivalence

- A number of useful laws can be derived from the first four
- It is vital to remember all of these when solving complex Boolean equations

Associative Law

- Some operators in math are *associative*
- For example: $(a + b) + c = a + (b + c)$
- Same applies to $\wedge$ and $\vee$

$$a \wedge (b \wedge c) \equiv (a \wedge b) \wedge c$$
$$a \vee (b \vee c) \equiv (a \vee b) \vee c$$

Absorption Law

- There is a special case of the Distributive Law where one variable is *absorbed* (i.e. eliminated)
- Applies to both $\wedge$ and $\vee$

$$a \wedge (a \vee b) \equiv a$$
$$a \vee (a \wedge b) \equiv a$$

Idempotent Law

- When a statement is combined with itself, it is equivalent to just the statement (no duplicate)
- This applies to both $\wedge$ and $\vee$

$$a \wedge a \equiv a$$
$$a \vee a \equiv a$$

Involution Law

- One of the most basic equivalences in logic is the *double negation*
- It is fairly obvious, so not more needs to be said

$$\neg \neg a \equiv a$$

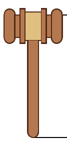
Domination Law

- This might look similar to the identity law, but look very careful at which operator is being used
- These will result in either true or false.

$$a \vee \text{true} \equiv \text{true}$$
$$a \wedge \text{false} \equiv \text{false}$$

DeMorgan's Law

- So, it states you can change the operator from ∧ to ∨ or vice-versa
- If you negate both operands



$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$
$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

61

Truth Table – Testing Not-Or

a	b	¬ a	¬ b	¬ a ∧ ¬ b	¬ (a ∨ b)
T	T	F	F	F	F
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

62

Truth Table – Testing Not-And

a	b	¬ a	¬ b	¬ a ∨ ¬ b	¬ (a ∧ b)
T	T	F	F	F	F
T	F	F	T	T	T
F	T	T	F	T	T
F	F	T	T	T	T

63

Example

- So, can we simplify the expression below?
- Let's create a truth table

$$(a \wedge b) \vee (\neg a \wedge b)$$

64

Boolean Algebra

- Truth tables become unwieldy as the number of variables increase
- Logical algebra is another way to evaluate equivalence
- Equivalences can be used to generate one expression from another

65

Example Simplification

$(a \wedge b) \vee (\neg a \wedge b) =$

$(a \vee \neg a) \wedge b =$ After using Distributive Law

$\text{true} \wedge b =$ After using Complement Law

b After using Identify Law

66

Implication

The operator of science

67

Implication

- The only Boolean operator that causes confusion is implication
- However, its usage is ***vital*** to understand – since it is used your programs (even if you might not see it)

68

Implication

- For "p implies q"…
- p is called the *antecedent* (or hypothesis or assumption)
- q is called the *consequent* (or conclusion)

69

Implication

- "p implies q" is contradicted (false) **only** when…

  p is true and **q is false**

- In all other cases, it is true

70

Standard Precedence Levels

1	¬
2	∧
3	∨ ⊕
4	→

Highest Level

Lowest Level

71

Implies Example

True

True

1 < 2 → 10 < 40

Result: True

72

Implies Example 2

True — False

`5 > 3 → 44 < 8`

Result: False

73

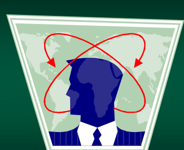
Implies Example 3

False — False

`10 < 2 → 12 < 10`

Result: True

74



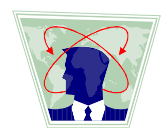
Analyzing Implication

Understanding is Power

75

Analyzing Implication

- Implication is both simple and complex
- It is used in all aspects of logical proof and the basis of all programs
- Understanding is complexity is essential to understanding logic (and discrete math)



76

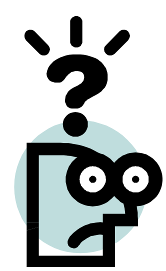
Implication Examples

- *If the Moon is made of cheese then the Moon is a tasty snack.*
- *If the flag has a bear then it's the Flag of California.*
- *If it is a fish then it is lives in water.*
- *If the university is Sacramento State then the mascot is a hornet.*

77

Many Ways to Say "Implies"

- *A implies B*
- *A → B*
- *B if A*
- *If A then B*
- *B given A*



78

Example From History

- Let's look at a implication from California history
- During the Gold Rush, people were inspired by a simple idea…
- *"If I pan for gold then I'll get rich!"*

79

"If I pan for gold then I'll get rich"

- Let's look at this statement closer
- It can be rewritten: *"Pan of Gold→ Get Rich"* or, very tersely, *"P → R"*

```
P → R
```


80

"If I pan for gold then I'll get rich"

- There four combinations of the truth table
- Which of these combinations would invalidate the statement

```
P → R
```


81

True → True

- If P is true, and R is true…
- *"We panned for gold and got rich"*
- Statement is true
  - we asserted that if P is true then R is true
  - since both are true, the statement is affirmed
  - true → true = true

82

False → True

- What if P is false and R is true
- *"We didn't pan for gold and got rich"*
- Statement is true
  - the fact we got rich (without panning for gold), doesn't mean that the statement is false
  - it has not contradicted the statement
  - false → true = true

83

False → False

- What if P is false and R is false?
- *"We didn't pan for gold and didn't get rich"*
- Statement is true
  - the fact that both are false, still does not contraction our original statement
  - it stated "IF we pan for gold **then** we get rich"
  - false → false = true

84

True → False

- Finally what if P is true and R is false?
- *We panned for gold, but didn't get rich*
- Statement is false
  - we asserted if P is true then R must be true
  - however, since this contradicts the assertion, the result of the implication is false
  - true → false = false

85

Implication Hiding in Plain Sight

- Consider the expression below
- The word "then" is alternative way of saying "implies"
- So, Is it True? False?

```
if x > 2 then x² > 4
```


86

Implication Hiding in Plain Sight

- There are different values of x that will make the antecedent and consequent both true and false
- If both are true, then the statement is correct

```
if x > 2 then x² > 4
```

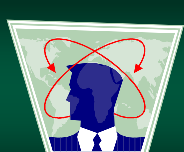

87

Implication Hiding in Plain Sight

- If **x > 2** is false, then we don't care about the consequent

```
if x > 2 then x² > 4
```


88



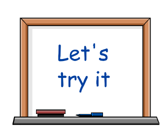
Deconstructing Implication

Other operators; same logic

89

Deconstructing Implication

- The implication logic can be broken down into the forms that are easier to remember
- This is actually quite important when we cover a few logical tricks later one
- So, let's look at the truth table for other operators



90

Analyzing Implication

- So, can implication be written using just logical "and", "or", or "not"?
- Yes, we can!

$p \rightarrow q$

Spring 2020 Sacramento State - Cook - CSc 28 91

91

Truth Table – One way to do it

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

True whenever q is true

True whenever p is false

Spring 2020 Sacramento State - Cook - CSc 28 92

92

Analyzing Implication

- So, we can definite p → q as "not p or q"
- Like before, let's prove in our truth table

$p \rightarrow q \equiv \neg p \vee q$

Spring 2020 Sacramento State - Cook - CSc 28 93

93

Truth Table – Or Logic

p	q	¬p	¬p ∨ q	p → q
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Spring 2020 Sacramento State - Cook - CSc 28 94

94

Truth Table – One way to look at it

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

Only false for:

$p \wedge \neg q$

We can negate this.

Spring 2020 Sacramento State - Cook - CSc 28 95

95

Analyzing Implication

- So, we can definite p → q as "not (p and not q)"
- It doesn't look quite right, let's test it out

$p \rightarrow q \equiv \neg(p \wedge \neg q)$

Spring 2020 Sacramento State - Cook - CSc 28 96

96

Truth Table – And Logic

p	q	¬ q	p ∧ ¬ q	¬ (p ∧ ¬ q)	p → q
T	T	F	F	T	T
T	F	T	T	F	F
F	T	F	F	T	T
F	F	T	F	T	T

97

Analyzing Implication

$$p \rightarrow q \equiv \neg (p \land \neg q)$$
$$\equiv \neg p \lor q$$

98