Predicates and Quantifiers

Announcements

Late Day Policy

Up to 6 late days to use during the quarter.
Lets you submit a homework up to 24 hours later than normal.
Max of 3 late days per assignment.

Just submit on gradescope, don't need to tell us you're using late days.

If you have extra late days after the last homework, we turn each late day into a "full credit concept check"
Logisitcally too much to count late days on the small checks; we assume you would have gotten full credit.

Announcements

Submitting HW1

Give yourself at least a few minutes to submit.

Gradescope asks you to "select pages" please do that!

Deadline is 10PM (not midnight)

~~The "feedback" question is now in gradescope quiz form. Please fill it out there.~~

On notation…

Logic is fundamental. Computer scientists use it in programs, mathematicians use it in proofs, engineers use it in hardware, philosophers use it in arguments,….

…so everyone uses different notation to represent the same ideas.

Since we don't know exactly what you're doing next, we're going to show you a bunch of them; but don't think one is "better" than the others!

Digital Circuits

Computing With Logic

T corresponds to **1** or "high" voltage

F corresponds to **0** or "low" voltage

Gates

Take inputs and produce outputs (functions)

Several kinds of gates

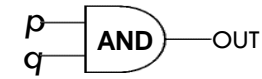Correspond to propositional connectives (most of them)
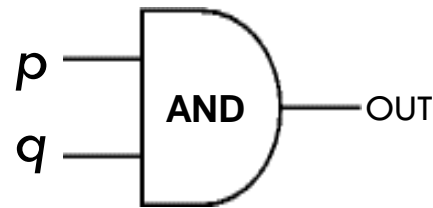
And Gate

AND Connective   vs.   **AND Gate**

$p \wedge q$

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

p	q	OUT
1	1	1
1	0	0
0	1	0
0	0	0

AND

p — AND — OUT
q —

"block looks like D of AND"

Or Gate

OR Connective   vs.   **OR Gate**

$p \vee q$

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F
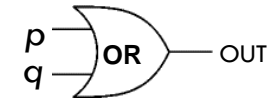
p	q	OUT
1	1	1
1	0	1
0	1	1
0	0	0
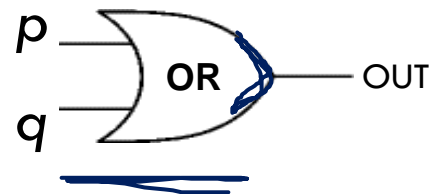
"arrowhead block looks like V"

Not Gates

NOT Connective   vs.   NOT Gate

$\neg p$

p —NOT⊳o— OUT
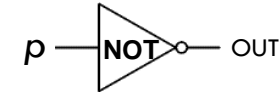
p	$\neg p$
T	F
F	T

p	OUT
1	0
0	1

Also called
inverter

p —NOT⊳o—OUT
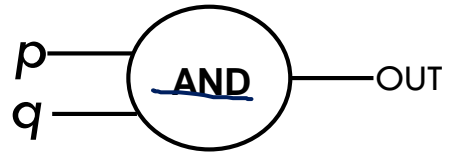
out

¬p
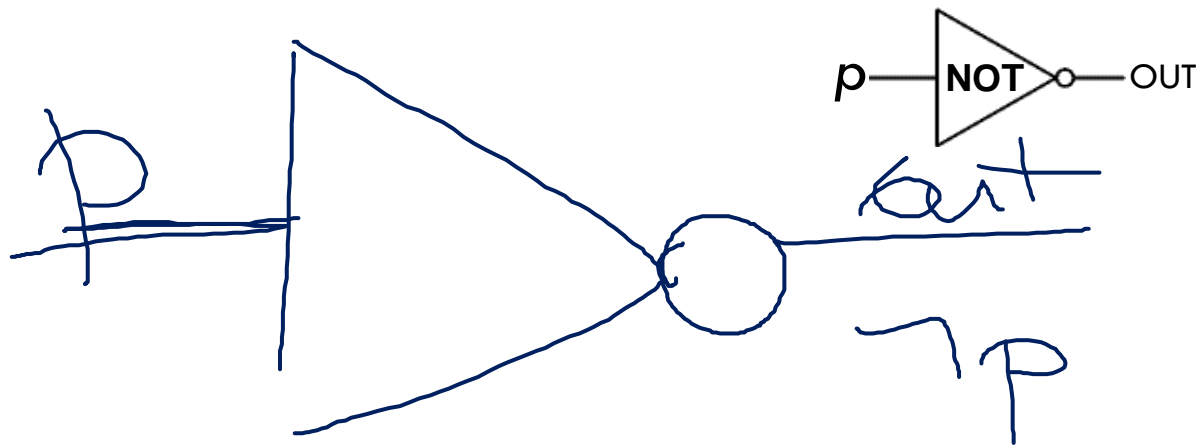
Blobs are Okay!

You may write gates using blobs instead of shapes!
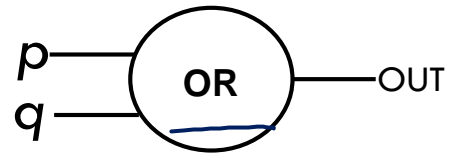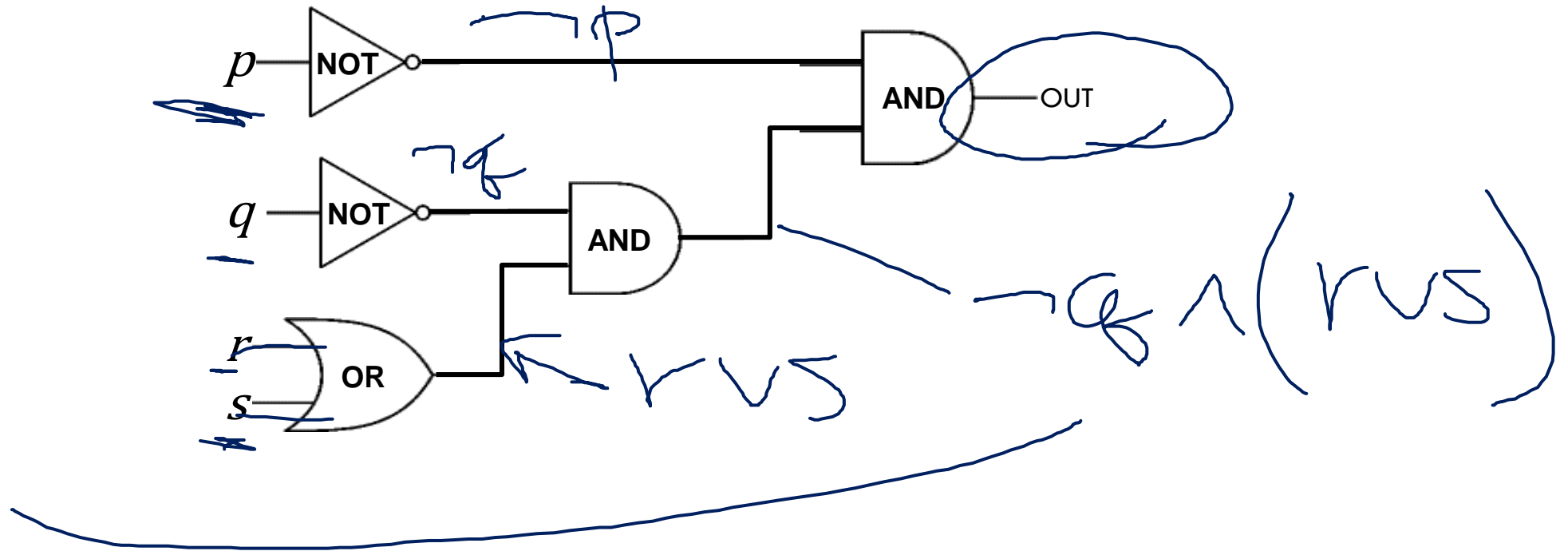
Combinational Logic Circuits

Values get sent along wires connecting gates
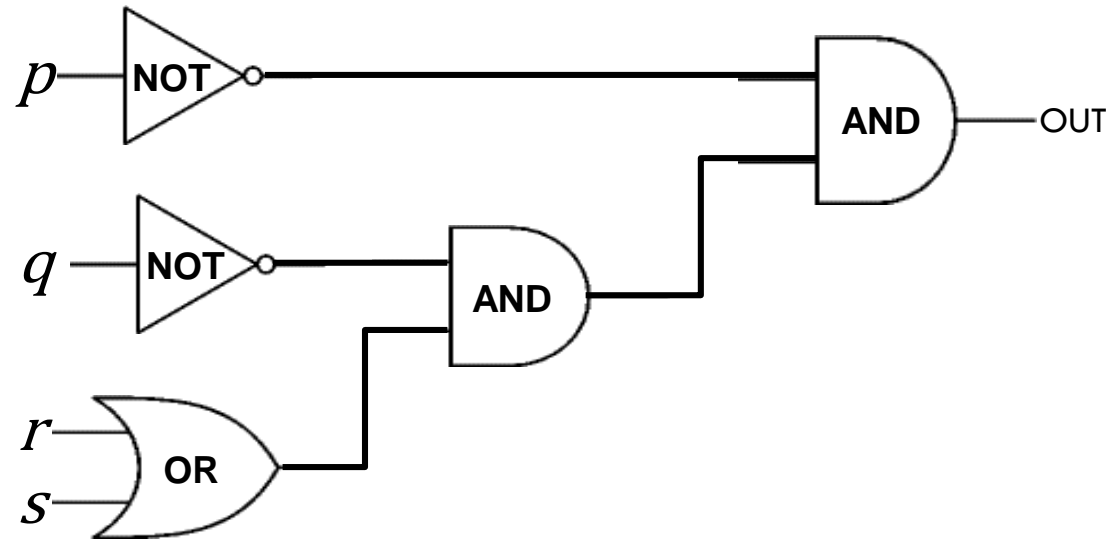
Combinational Logic Circuits

$$\neg p \wedge (\neg q \wedge (r \vee s))$$

Values get sent along wires connecting gates

Combinational Logic Circuits
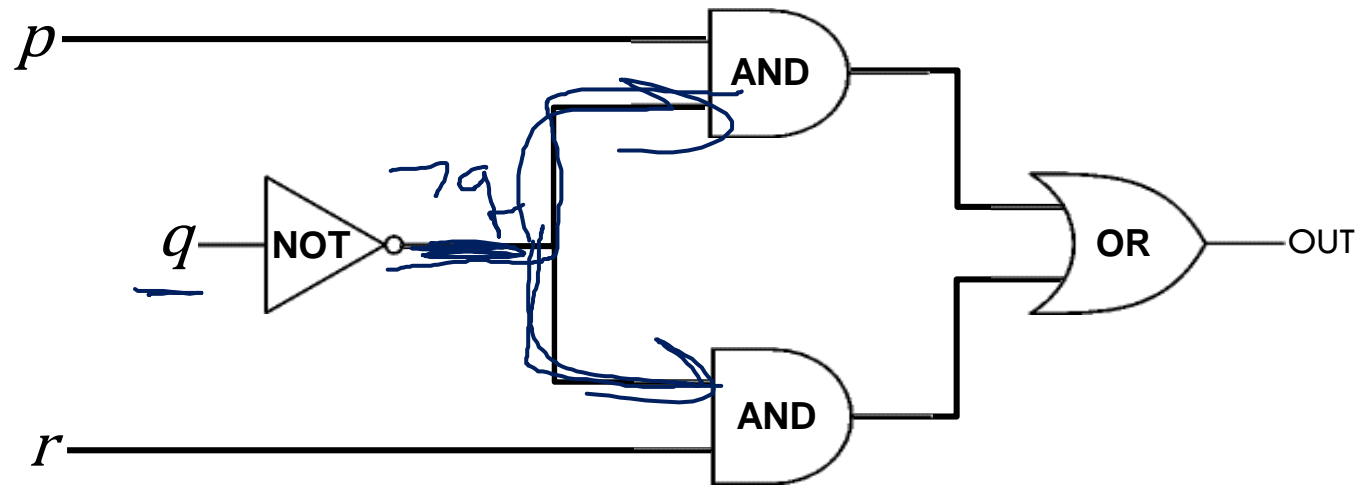
Wires can send one value to multiple gates!

Combinational Logic Circuits
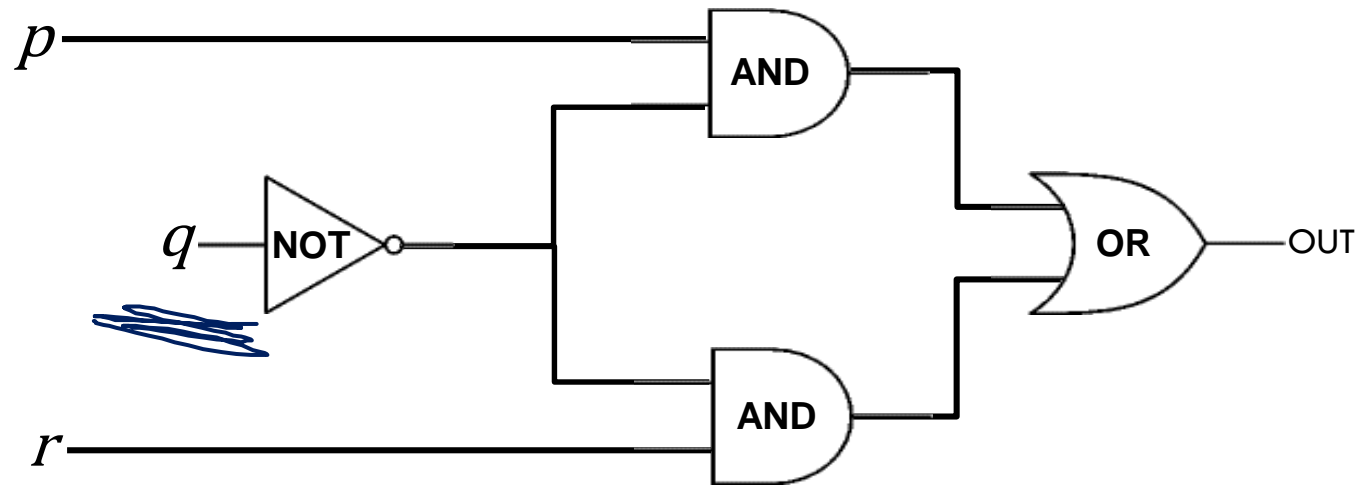
Wires can send one value to multiple gates!

$$(p \wedge \neg q) \vee (\neg q \wedge r)$$

More Practice

Some quarters, we spend about half-a-lecture building out a circuit to represent a moderately complicated function.

It's 50% a way to practice with the different notation, 50% fun historical context (people used to build circuits by-hand a lot. It's less common now – decent programs exist to do it now).

If you want to get that extra practice, I'll post an extra video with that content.

But also have practice problems on section handout if you would rather get the content that way.

There are no new principles/ideas here. Only new ways of representing them.

Boolean Algebra

Meet Boolean Algebra

Preferred by some mathematicians and circuit designers.

"or" is $+$

"and" is $\cdot$ (i.e. "multiply")

"not" is ' (an apostrophe after a variable)

Why?

Mathematicians like to study "operations that work kinda like 'plus' and 'times' on integers."

Circuit designers have a lot of variables, and this notation is more compact.

Meet Boolean Algebra
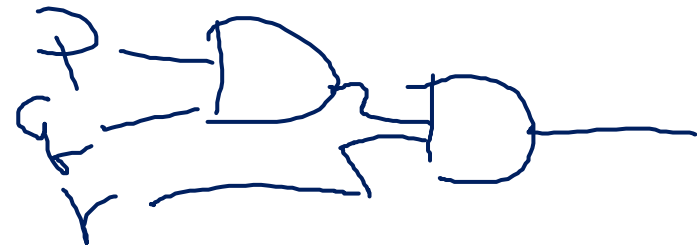
Name	Variables	"True/False"	"And"	"Or"	"Not"	Implication
Java Code	`boolean b`	`true,false`	`&&`	`||`	`!`	No special symbol
Propositional Logic	p, q, r	`T, F`	$\wedge$	$\vee$	$\neg$	$\rightarrow$
Circuits	Wires	1, 0	And	Or	Not	No special symbol
Boolean Algebra	a, b, c	1,0	· ("multiplication")	+ ("addition")	' (apostrophe after variable)	No special symbol
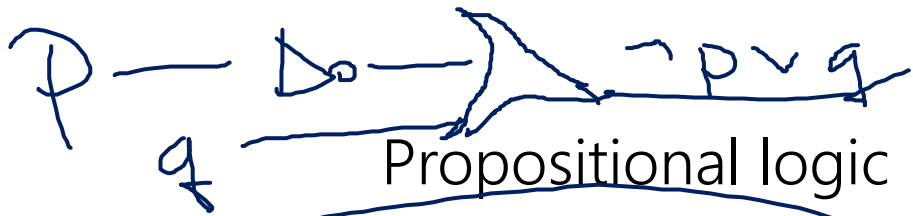
Propositional logic

$$(p \wedge q \wedge r) \vee s \vee \neg t$$

Boolean Algebra

$$pqr + s + t'$$

Comparison

Propositional logic

$$(p \land q \land r) \lor s \lor \neg t$$

Boolean Algebra

$$pqr + s + t'$$

Remember this is just an alternate notation for the same underlying ideas.

So that big list of identities? Just change the notation and you get another big list of identities!

Sometimes names are different ("involution" instead of "double negation"), but the core ideas are the same.

Boolean Algebra

$P \lor F = P$

Axioms

Closure
$a + b$ is in $\mathbb{B}$
$a \bullet b$ is in $\mathbb{B}$

Commutativity
$a + b = b + a$
$a \bullet b = b \bullet a$

Associativity
$a + (b + c) = (a + b) + c$
$a \bullet (b \bullet c) = (a \bullet b) \bullet c$

Identity
$a + 0 = a$
$a \bullet 1 = a$

Distributivity
$a + (b \bullet c) = (a + b) \bullet (a + c)$
$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$

Complementarity
$a + a' = 1$
$a \bullet a' = 0$

Boolean Algebra

$P \vee P = P$

Theorems

Null

$$X + 1 = 1$$
$$X \bullet 0 = 0$$

Idempotency

$$X + X = X$$
$$X \bullet X = X$$

Involution

$$(X')' = X$$

Uniting

$$X \bullet Y + X \bullet Y' = X$$
$$(X + Y) \bullet (X + Y') = X$$

Boolean Algebra

Absorbtion

$$X + X \bullet Y = X$$
$$(X + Y') \bullet Y = X \bullet Y$$
$$X \bullet (X + Y) = X$$
$$(X \bullet Y') + Y = X + Y$$

DeMorgan

$$(X + Y + \cdots)' = X' \bullet Y' \bullet \cdots$$
$$(X \bullet Y \bullet \cdots)' = X' + Y' + \cdots$$

Consensus

$$(X \bullet Y) + (Y \bullet Z) + (X' \bullet Z) = X \bullet Y + X' \bullet Z$$
$$(X + Y) \bullet (Y + Z) \bullet (X' + Z) = (X + Y) \bullet (X' + Z)$$

Factoring

$$(X + Y) \bullet (X' + Z) = X \bullet Z + X' \bullet Y$$
$$X \bullet Y + X' \bullet Z = (X + Z) \bullet (X' + Y)$$

A Few Fun Facts

That you're not responsible for:

The identities are divided into "axioms" and "theorems"

Mathematicians (and some computer scientists, like me ☺) will sometimes study what minimum starting points ("the axioms") will be enough to derive all the usual facts we rely on ("the theorems")
That's what I meant by "operations that work kinda like plus and times"

For our purposes, we won't make a distinction here, but we will use similar thinking later in the course.

Boolean algebra makes things like commutativity axioms (starting points, things we assume) with propositional logic, we start from the truth tables and can derive that commutativity is true. For this class, though, it's a fact you can use either way.

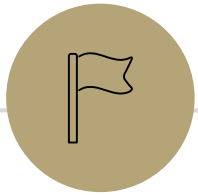# Why ANOTHER way of writing down logic?

This is the third one!?

Because, in your future courses, you'll use any/all of them.

Remember there aren't new concepts here, just new representations.

We mostly use propositional notation (∧,∨, ¬,etc.) but we'll use them all a bit so you're ready for any of them in your future courses.

Practice in section and on homework.

Canonical Forms

Back to the old notation.

Canonical Forms

A truth table is a unique representation of a Boolean Function.
If you describe a function, there's only one possible truth table for it.

Given a truth table you can find many circuits and many compound prepositions to represent it.
Think back to when we were developing the law of implication...

It would be nice to have a "standard" proposition (or standard circuit) we could always write as a starting point.
So we have a (possibly) shorter way of telling if we have the same function.
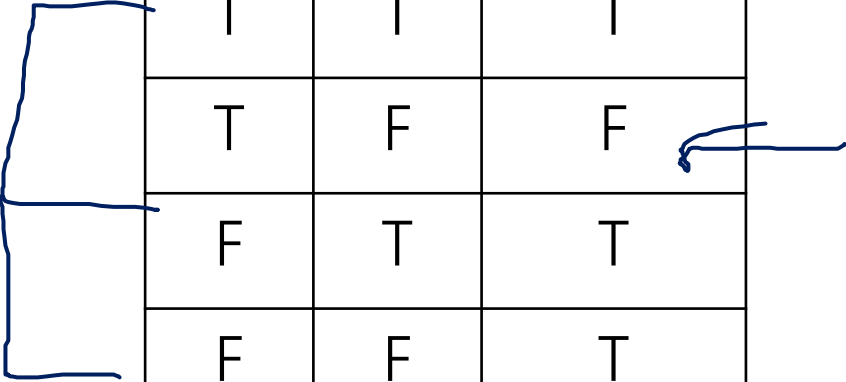
Using Our Rules

WOW that was a lot of rules.

Why do we need them? Simplification!

Let's go back to the "law of implication" example.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

When is the implication true? Just "or" each of the three "true" lines!

$$(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q)$$

Also seems pretty reasonable

So is $(p \wedge q) \vee (\neg p \wedge q) \vee (\neg p \wedge \neg q) \equiv (\neg p \vee q)$

i.e. are these both alternative representations of $p \rightarrow q$?

Disjunctive Normal Form (DNF)

a.k.a. OR of ANDs

a.k.a Sum-of-Products Form

a.k.a. Minterm Expansion

1. Read the true rows of the truth table
2. AND together all the settings in a given (true) row.
3. OR together the true rows.

Disjunctive Normal Form

p	q	$G(p, q)$
T	T	T
T	F	F
F	T	T
F	F	F

$p \wedge q$

$\neg p \wedge q$

1. Read the true rows of the truth table
2. AND together all the settings in a given (true) row.
3. OR together the true rows.

$$G(p, q) \equiv (p \wedge q) \vee (\neg p \wedge q)$$

Another Canonical Form

DNF is a great way to represent functions that are usually false.
If there are only a few true rows, the representation is short.

What about functions that are usually true?

Well G is equivalent to $\neg\neg G$, and $\neg G$ is a function that is usually false.

Let's try taking the Sum-of-Products of $\neg G$ and negating it.

Another Canonical Form

p	q	$G(p,q)$	$\neg G(p,q)$
T	T	T	F
T	F	F	T
F	T	T	F
F	F	F	T

$p \wedge \neg q$

$\neg p \wedge \neg q$

1. Read the true rows of the truth table
2. AND together all the settings in a given (true) row.
3. OR together the true rows.

$$\neg G(p,q) \equiv (p \wedge \neg q) \vee (\neg p \wedge \neg q)$$
$$G(p,q) \equiv \neg[(p \wedge \neg q) \vee (\neg p \wedge \neg q)]$$
$$G(p,q) \equiv [\neg(p \wedge \neg q) \wedge \neg(\neg p \wedge \neg q)]$$
$$G(p,q) \equiv [(\neg p \vee q) \wedge (p \vee q)]$$

This is not in Disjunctive Normal Form! It's something else, though…

Conjunctive Normal Form

a.k.a. AND of ORs

a.k.a. Product-of-Sums Form

a.k.a. Maxterm Expansion

1. Read the false rows of the truth table
2. OR together the negations of all the settings in the false rows.
3. AND together the false rows.

Or take the DNF of the negation of the function you care about, and distribute the negation.

Normal Forms

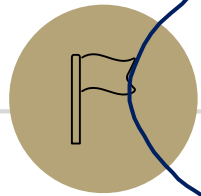Don't simplify any further! Don't factor anything out (even if you can). The point of the canonical form is we know exactly what it looks like, you might simplify differently than someone else.

Why? Easier to understand for people.

Inside the parentheses are only ORs between the parentheses are only ANDs (or vice versa).

You'll use these more in later courses.

$$x + 2 = 5$$

Predicates!

Predicate Logic

So far our propositions have worked great for fixed objects.

What if we want to say "If $x > 10$ then $x^2 > 100$."

$x > 10$ isn't a proposition. Its truth value depends on x.

We need a function that can take in a value for x and output True or False as appropriate.

Predicates

A function that outputs true or false.

`Cat(x):=` "x is a cat"

`Prime(x) :=` "x is prime"

`LessThan(x,y):=` "x<y"

`Sum(x,y,z):=` "x+y=z"

`HasNChars(s,n):=` "string `s` has length `n`"

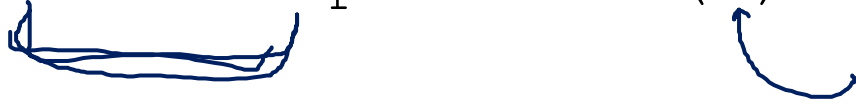Numbers and types of inputs can change. Only requirement is output is Boolean.

Analogy

Propositions were like Boolean variables.

What are predicates? Functions that return Booleans

```
public boolean predicate(…)
```

Translation

Translation works a lot like when we just had propositions.

Let's try it…

x is prime or x^2 is odd or $x = 2$.

$$\text{Prime}(x) \vee \text{Odd}(x^2) \vee \text{Equals}(x, 2)$$

Domain of Discourse

x is prime or x^2 is odd or $x = 2$.

$\mathtt{Prime}(x) \lor \mathtt{Odd}(x^2) \lor \mathtt{Equals}(x, 2)$

Can x be 4.5? What about "abc" ?

I never intended you to plug 4.5 or "abc" into x.

When you read the sentence you probably didn't imagine plugging those values in....

Domain of Discourse

x is prime or x^2 is odd or $x = 2$.

$\mathrm{Prime}(x) \lor \mathrm{Odd}\left(x^2\right) \lor \mathrm{Equals}(x, 2)$

To make sure we **can't** plug in 4.5 for x, predicate logic requires deciding on the types we'll allow

Domain of Discourse

The *types* of inputs allowed in our predicates.

Try it...

What's a possible domain of discourse for these lists of predicates?

1. "x is a cat", "x barks", "x likes to take walks"

2. "x is prime", "$x=5$" "$x < 20$" "x is a power of two"

3. "x is enrolled in course y", "y is a pre-req for z"

Try it...

What's a possible domain of discourse for these lists of predicates?

1. "x is a cat", "x barks", "x likes to take walks"

   "Mammals", "pets", "dogs and cats", ...

2. "x is prime", "$x=5$" "$x<20$" "x is a power of two"

   "positive integers", "integers", "numbers", ...

3. "x is enrolled in course y", "y is a pre-req for z"

   "objects in the university course enrollment system", "university entities", "students and courses", ...

More than one domain of discourse might be reasonable...if it might affect the meaning of the statement, we specify it.

Quantifiers

Now that we have variables, let's really use them…

We tend to use variables for two reasons:

1. The statement is true for every x, we just want to put a name on it.
2. There's some x out there that works, (but I might not know which it is, so I'm using a variable).

Quantifiers

We have two extra symbols to indicate which way we're using the variable.

1. The statement is true for every x, we just want to put a name on it.

$\forall x \, (p(x) \wedge q(x))$ means "for every x in our domain, $p(x)$ and $q(x)$ both evaluate to true."

2. There's some x out there that works, (but I might not know which it is, so I'm using a variable).

$\exists x (p(x) \wedge q(x))$ means "there is an x in our domain, such that $p(x)$ and $q(x)$ are both true.

Quantifiers

$\forall x$ for all

We have two extra symbols to indicate which way we're using the variable.

1. The statement is true for every x, we just want to put a name on it.

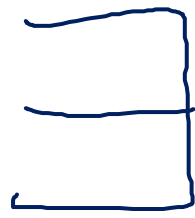$\forall x \, (p(x) \wedge q(x))$ means "for every x in our domain, $p(x)$ and $q(x)$ both evaluate to true."

Universal Quantifier

"$\forall x$"

"for each x", "for every x", "for all x" are common translations
Remember: upside-down-A for All.

Quantifiers

∃ exists

2. There's some x out there that works, (but I might not know which it is, so I'm using a variable).

$\exists x(p(x) \wedge q(x))$ means "there is an x in our domain, for which $p(x)$ and $q(x)$ are both true.

Translations

"For every x, if x is even, then $x = 2$."

"There are x, y such that $x < y$."

$\exists x \, (\text{Odd}(x) \wedge \text{LessThan}(x, 5))$

$\forall y \, (\text{Even}(y) \wedge \text{Odd}(y))$

pollev.com/robbie

Help me adjust my explanation!

Translations

"For every x, if x is even, then $x = 2$."

$$\forall x (\text{Even}(x) \rightarrow \text{Equal}(x, 2))$$

"There are x, y such that x $< y$."

$$\exists x \exists y (\text{LessThan}(x, y))$$

$$\exists x (\text{Odd}(x) \wedge \text{LessThan}(x, 5))$$

There is an odd number that is less than 5.

$$\forall y (\text{Even}(y) \wedge \text{Odd}(y))$$

All numbers are both even and odd.

Translations

More practice in section and on homework.

Also a reading on the webpage –

An explanation of why "for any" is not a great way to translate ∀ (even though it looks like a good option on the surface)

More information on what happens with multiple quantifiers (we'll discuss more on Wednesday).

Evaluating Predicate Logic

"For every x, if x is even, then $x = 2$." / $\forall x(\texttt{Even}(x) \rightarrow \texttt{Equal}(x, 2))$

Is this true?

Evaluating Predicate Logic

"For every x, if x is even, then $x = 2$." / $\forall x (\texttt{Even}(x) \rightarrow \texttt{Equal}(x, 2))$

Is this true?

TRICK QUESTION! It depends on the domain.

Prime Numbers	Positive Integers	Odd integers
True	False	True (vacuously)

One Technical Matter

How do we parse sentences with quantifiers?
What's the "order of operations?"

We will usually put parentheses right after the quantifier and variable to make it clear what's included. If we don't, it's the rest of the expression.

Be careful with repeated variables...they don't always mean what you think they mean.

$\forall x \big(P(x) \big) \land \forall x (Q(x))$ are different x's.

Bound Variables

What happens if we repeat a variable?

Whenever you introduce a new quantifier with an already existing variable, it "takes over" that name until its expression ends.

$$\forall x (P(x) \land \forall x [Q(x)] \land R(x))$$

It's common (albeit somewhat confusing) practice to reuse a variables when it "wouldn't matter".

Never do something like the above: where a single name switches from gold to purple back to gold. Switching from gold to purple only is usually fine…but names are cheap.

More Practice

Let your domain of discourse be fruits. Translate these

There is a fruit that is tasty and ripe.

For every fruit, if it is not ripe then it is not tasty.

There is a fruit that is sliced and diced.

More Practice

Let your domain of discourse be fruits. Translate these

There is a fruit that is tasty and ripe.

$$\exists x(\texttt{Tasty}(x) \wedge \texttt{Ripe}(x))$$

For every fruit, if it is not ripe then it is not tasty.

$$\forall x(\neg \texttt{Ripe}(x) \rightarrow \neg \texttt{Tasty}(x))$$

There is a fruit that is sliced and diced.

$$\exists x(\texttt{Sliced}(x) \wedge \texttt{Diced}(x))$$