

# CS 3 2009 Lecture 2

Today:

- Quick review of last time.
- Extending our logic: predicate calculus.

# Propositional Calculus Notation

- We use compact notation for the connectives:

C	Usage	Sym.	Op.
not	not $p$	$\neg p$	negation
and	$p$ and $q$	$p \wedge q$	conjunction
or	$p$ or $q$	$p \vee q$	disjunction
if	$p$ implies $q$	$p \Rightarrow q$	implication
iff	$p$ equiv. to $q$	$p = q$	equivalence

(iff means “if and only if”)

“CS 3 is graded pass/fail *and* L.A. does not have clean air.”  
 $= p \wedge \neg q$

- Precedence: (highest)  $\neg \wedge \vee \Rightarrow =$  (lowest)

For example:

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

Use parentheses when appropriate.

# Truth tables

Since all that matters is the truth or falsehood of the connected propositions, we can simply enumerate all the possibilities. (T = true; F = false).

$p$	$q$	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p = q$
F	F	T	F	F	T	T
F	T	T	F	T	T	F
T	F	F	F	T	F	F
T	T	F	T	T	T	T

- Connectives mostly mean the same as in English.
- What about  $p \Rightarrow q$  (antecedent  $\Rightarrow$  consequent)?

“If it rains, the picnic is cancelled.”

it rains  $\Rightarrow$  the picnic is cancelled

And if it doesn't rain? (What if it snows?)

What about:

It rains  $\Rightarrow 2 + 3 = 5$

# Basic properties

We can immediately deduce some properties of the connectives:

- Commutativity of disjunction and conjunction:

$$x \vee y = y \vee x$$

$$x \wedge y = y \wedge x$$

- Associativity of disjunction and conjunction:

$$(x \vee y) \vee z = x \vee (y \vee z)$$

$$(x \wedge y) \wedge z = x \wedge (y \wedge z)$$

(So the parentheses are unnecessary!)

- Distributivity of disjunction over conjunction and of conjunction over disjunction:

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

(Note difference from integer addition and multiplication: works both ways here.)

## More properties

We could write infinitely many properties.

- Some of the more useful are:

$$\neg\neg x = x$$

$$x \wedge x = x$$

$$x \vee x = x$$

$$x \vee \neg x = \mathbf{true} \quad (\text{excluded middle})$$

$$x \wedge \neg x = \mathbf{false}$$

$$x \vee \mathbf{true} = \mathbf{true}$$

$$x \wedge \mathbf{true} = x$$

$$x \vee \mathbf{false} = x$$

$$x \wedge \mathbf{false} = \mathbf{false}$$

$$x \wedge (x \vee y) = x$$

$$x \vee (x \wedge y) = x$$

$$x \vee \neg x \wedge y = x \vee y \quad (\text{absorption})$$

## More

De Morgan's laws:

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

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

Inference rules:

$$p \Rightarrow p \vee q \quad (\text{addition})$$

$$p \wedge q \Rightarrow p \quad (\text{simplification})$$

$$(p \vee q) \wedge \neg p \Rightarrow q \quad (\text{disjunctive syllogism})$$

$$(p \Rightarrow q) \wedge p \Rightarrow q \quad (\text{modus ponens})$$

$$(p \Rightarrow q) \wedge \neg q \Rightarrow \neg p \quad (\text{modus tollens})$$

$$(p \Rightarrow \neg p) \Rightarrow \neg p \quad (\text{reductio ad absurdum})$$

$$p \Rightarrow q = \neg q \Rightarrow \neg p \quad (\text{contraposition})$$

Watch out for:

- Fallacy of affirming the consequent:

$$(p \Rightarrow q) \wedge q \not\Rightarrow p$$

- Fallacy of denying the antecedent:

$$(p \Rightarrow q) \wedge \neg p \not\Rightarrow q$$

# Inference Rules

The “inference rules” are also often written similarly to syllogisms:

- Modus ponens:

$$\begin{array}{rcl} p \Rightarrow q & \textit{Premise} & \\ p & \textit{Premise} & \\ \hline \therefore q & \textit{Conclusion} & \end{array}$$

- Modus tollens:

$$\begin{array}{rcl} p \Rightarrow q & \textit{Premise} & \\ \neg q & \textit{Premise} & \\ \hline \therefore \neg p & \textit{Conclusion} & \end{array}$$

Or as:

$$\begin{array}{rcl} p \Rightarrow q , p & \textit{Premises} & \\ \hline q & \textit{Conclusion} & \end{array}$$

# Why use inference rules

Why should we use inference rules?

- In algebra, it's obvious: there are infinitely many integers, and even more real numbers.
- But we are only dealing with two values: **true** and **false**.
- Consider proving, using truth tables, that

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

for all values of  $a, b, c, d \dots$

How about proving that

$$\neg(a \wedge b \wedge c \wedge d \wedge e) = \neg a \vee \neg b \vee \neg c \vee \neg d \vee \neg e$$

<b>C</b>	<b>Usage</b>	<b>Sym.</b>	<b>Op.</b>
not	not $p$	$\neg p$	negation
and	$p$ and $q$	$p \wedge q$	conjunction
or	$p$ or $q$	$p \vee q$	disjunction
if	$p$ implies $q$	$p \Rightarrow q$	implication
iff	$p$ equiv. to $q$	$p = q$	equivalence

# Summary

Summary of propositional calculus:

- Statements/literals/atoms that are either **true** or **false**.
- Connectives:  $\neg$   $\wedge$   $\vee$   $\Rightarrow$   $=$
- Tautologies/analytic propositions: always **true**, can be used for inference.
- Contradictions: always **false**.
- Synthetic propositions: the rest; either **true** or **false** depending on the circumstances.
- Can always check if a statement is a tautology, contradiction, or a synthetic proposition by making the truth table.
- Inference rules are easier to use than truth tables for large expressions.

# Predicates

Propositional calculus deals only with propositions that are **true** or **false** without saying more about what happens “inside” the propositions.

- This isn't powerful enough for dealing with programs.

*Example.* GCD program:

```
WHILE x # y DO
  IF x > y THEN
    x := x - y
  ELSE
    y := y - x
  END
END;
```

At the end of the loop (if the program gets that far!), we must have that  $x=y$ ; not a statement in propositional calculus because  $x$  and  $y$  are *integers*, not propositions.

# Introducing Predicates

Instead of dealing just with propositions, we'll introduce “atoms” (atomic boolean expressions). We'll admit pretty much anything that evaluates to **true** or **false**.

- A statement that says some entity has some property or that some entities have some property we call a *predicate*.

$$x > 0$$

$$x = y$$

$$y = 2x + 1$$

It's raining.

John is taller than Mary.

- Predicates can constrain either a single object or several.
- A predicate is a boolean function; a proposition is a boolean constant. (The distinction is not usually that important.)
- If the specifics of a predicate aren't important we can write it as  $P(x)$  or  $Q(x)$ .

# Strong and Weak

In logic, we talk about “strong” and “weak” statements in a very specific way.

- Strong statement: class ends at 1:53 p.m.
- Not so strong statement: class ends sometime this afternoon.
- Weak statement: class ends, or it doesn't.
- We say that  $p$  is stronger than  $q$  if and only if

$$p \Rightarrow q$$

- If  $P$  and  $Q$  are predicates, then we say that  $P$  is stronger than  $Q$  if and only if

$$P(x) \Rightarrow Q(x)$$

for all possible values of  $x$ .

- Often written (Dijkstra)

$$[P \Rightarrow Q]$$

## Examples

- $x = 1$  is a stronger statement than  $x > 0$ .
- $x > 1$  is a stronger statement than  $x > 0$ .
- A stronger statement is **true** for “strictly fewer” states than a weaker statement. (Note that the number of states we allow is often infinite.)
- The weakest statement is

**true**

It’s always true! Furthermore, for any other statement  $p$ , it’s always the case that  $p \Rightarrow \mathbf{true}$ ; so  $p$  is stronger than **true**.

- Conversely, the strongest statement is...?
- $x = 1$  is neither weaker nor stronger than  $x = 2$ .
- $x = 1$  is neither weaker nor stronger than  $x > 2$ .

## Important remark

Given a predicate  $P$ ,

$$P \wedge Q \Rightarrow P$$

for all  $Q$ ; similarly,

$$P \Rightarrow P \vee Q.$$

So:

- We *strengthen* a statement  $P$  by replacing it by  $P \wedge Q$ .
- We *weaken* a statement  $P$  by replacing it by  $P \vee Q$ .

# Universal Quantifier

We've already informally talked about the “number of states”; “for all  $x \dots$ ”; and “if there exists. . .”

These constructs are used so frequently we introduce notation for them and rules for calculating with them. (We'll use “Dijkstra notation.”)

- “For all  $x$  such that  $x$  is a natural number, it is not the case that  $x^2$  equals -1.” (A true statement.)

$$\forall x : x \in \mathcal{N} : x^2 \neq -1$$

- $\forall$  is pronounced “forall” and is called the *universal quantifier*.
- If it's obvious what the range of  $x$  is (or if it is otherwise explicit, or if we're being sloppy), we can leave the range out:

$$\forall p :: \mathbf{false} \Rightarrow p$$

$$\forall p :: p \Rightarrow \mathbf{true}$$

# Existential Quantifier

- “There exists an  $x$  that is a natural number such that  $x^2$  equals 25.” (A true statement)

$$\exists x : x \in \mathcal{N} : x^2 = 25$$

- $\exists$  is pronounced “exists” and is called the *existential quantifier*.
- Note that there’s nothing magical about  $\forall$  and  $\exists$ , even if they look funny. For finite domains, they are equivalent to conjunctions and disjunctions:

$$\forall x : x \in \{0, 1, 2\} : x^2 < 5$$

$$(0^2 < 5) \wedge (1^2 < 5) \wedge (2^2 < 5)$$

$$\exists x : x \in \{0, 1, 2\} : x^2 = 4$$

$$(0^2 = 4) \vee (1^2 = 4) \vee (2^2 = 4)$$

## Other Quantifiers

Other quantifiers appear from time to time:

- *Counting quantifier, N*: “There are three natural numbers whose squares are smaller than 10.”

$$(Nx : x \in \mathcal{N} : x^2 < 10) = 3$$

- “There are five odd numbers smaller than 10.”

$$(Nx : x \in \mathcal{N} : (\exists y : y \in \mathcal{N} : x = 2y + 1)) = 5$$

You can think of others that are occasionally useful...

# Free and Bound

The quantifiers have introduced a major subtlety: the concept of *free* vs. *bound* variables.

- Really nothing new:  $x$  is bound (a “dummy” variable), and  $a$  is free, in

$$\int_0^a x^2 dx.$$

- Same idea here:  $x$  bound,  $a$  free:

$$\exists x :: x^2 = a$$

- Allowing expressions like

$$x \vee (\exists x :: x^2 = 0)$$

leads to huge difficulties. They can be overcome, but this isn't a class about formal logic; please just don't do it! Compare to:

$$\int_0^x x^2 dx$$

## Funny Empty Ranges

- “There exists a living dinosaur that is red.”

$\exists d : d \text{ is a living dinosaur} : d \text{ is red}$

This statement is **false**.

- “Every living dinosaur is red.”

$\forall d : d \text{ is a living dinosaur} : d \text{ is red}$

Hmm...

- “Every dinosaur has a skeleton.”

$D = \forall d : d \text{ is a dinosaur} : d \text{ has a skeleton}$

$DD = \forall d : d \text{ is a dead dinosaur} : d \text{ has a skeleton}$

$LD = \forall d : d \text{ is a living dinosaur} : d \text{ has a skeleton}$

$$D \stackrel{?}{=} DD \wedge LD$$

# Funny Empty Ranges

- Existential quantification over empty range always **false**.
- Universal quantification over empty range always **true**.
- If every  $x$  has property  $P$ , then there exists an  $x$  that has property  $P$ :

$$(\forall x :: P(x)) \Rightarrow (\exists x :: P(x))$$

Not a tautology!

*Example.* “If every living dinosaur is red, then there exists a living dinosaur that is red.”

## Inference rules for $\forall$ and $\exists$

The inference rules for  $\forall$  and  $\exists$  are the same as for  $\wedge$  and  $\vee$ :

$$(\forall x : x \in X : P(x)) \wedge a \in X \Rightarrow P(a)$$

Often easier to rewrite:

$$(\forall x : Q(x) : P(x))$$

$$(\forall x :: Q(x) \Rightarrow P(x))$$

$$P(a) \Rightarrow (\exists x :: P(x))$$

(Pretty obvious.)

Inversion: generalization of De Morgan's laws:

$$\neg(\forall x :: P(x)) = \exists x :: \neg P(x)$$

$$\neg(\exists x :: P(x)) = \forall x :: \neg P(x)$$

# Summary

- Propositional calculus.
- Atomic boolean expressions (“atoms”).
- “Strong” and “weak” statements.
- Quantifiers.
  
- Next time: applying what we’ve learned to programming.