

# Math 131 notes

Jason Riedy

8 September, 2008

## Contents

<b>1</b>	<b>Language of logic</b>	<b>1</b>
<b>2</b>	<b>Symbolic logic</b>	<b>3</b>
<b>3</b>	<b>Logical operators and truth tables</b>	<b>3</b>
<b>4</b>	<b>Properties of logical operators</b>	<b>5</b>
<b>5</b>	<b>Truth tables and logical expressions</b>	<b>6</b>
5.1	De Morgan's laws . . . . .	6
5.2	Logical expressions from truth tables . . . . .	7
<b>6</b>	<b>Conditionals</b>	<b>8</b>
6.1	English and the operator $\rightarrow$ . . . . .	8
6.2	Defining $p \rightarrow q$ . . . . .	9
6.3	Converse, inverse, and contrapositive . . . . .	9
6.4	If and only if, or $\leftrightarrow$ . . . . .	10
<b>7</b>	<b>Quantifiers</b>	<b>10</b>
7.1	Negating quantifiers . . . . .	11
7.2	Nesting quantifiers . . . . .	11
7.3	Combining nesting with negation . . . . .	12
<b>8</b>	<b>Logical deduction: Delayed until after the test</b>	<b>13</b>
<b>9</b>	<b>Homework</b>	<b>16</b>

*Notes also available as PDF.*

## 1 Language of logic

Goals:

- Determine when statements are **logical statements**.
- Recognize and transform between equivalent logical statements.
- Negate logical statements and quantified logical statements correctly.
- Apply the logical rules of deduction and follow *if-then* chains formally.

We start with definitions:

**logical statement** A declaration that is either true or false but not both. For example:

Today is Monday.

Languages contain many statements and declarations that are not logical statements:

Symbolic logic is fun.

While that is a declaration, it is neither true nor false *in general*. An example of a logical statement from set theory,

$x \in A$ .

**negation** A logical statement over the same topics that is false if the original statement is true or true if the original is false. The statement

My dog has fleas.

has as its negation

My dog does not have fleas.

However, a statement about my cat(s) cannot be a negation of either of the above regardless of which statements are true or false.

**quantifier** When a statement applies to *all*, *some*, *every* of something, the statement is **quantified**. The word denoting how many is the **quantifier**. This is where negation becomes tricky. For example, the statement

All dogs have fleas.

has as its negation

Some dogs do not have fleas.

Its negation is **not**

All dogs do not have fleas.

## 2 Symbolic logic

Expressing logical statements with symbols lets us focus on manipulating the logic itself. We can talk about the dog having fleas without mentioning dogs or fleas.

Variables like  $p$  and  $q$  can take the values true or false. Like many items, true and false are given different symbols by different authors. Common symbols include

true	false
T	F
1	0
⊤	⊥

I will use 1 and 0.

## 3 Logical operators and truth tables

The first operator is **negation**. This is a **unary** operation; it applies to a single operand. Two symbols are commonly used for the negation operator,  $\neg$  and  $\sim$ , as is adding a bar to a variable,  $\bar{p}$ . I will stick with the symbol  $\neg$  for most cases.

With one operand, listing all possible inputs and outputs is simple. The list is also called a **truth table**. The truth table for  $\neg$  is as follows:

$p$	$\neg p$
1	0
0	1

Programming languages may represent  $\neg$  with operators or functions like `!`, `not()`, `.NOT.`, `not`.

When **and** joins pieces of a logical statement, we understand that the statement as a whole is true only if both pieces are true. This is the **conjunction** operator. The *conjunction* operator  $\wedge$  is the symbol we use to represent the same idea. The truth table for the  $\wedge$  operator has four lines and is as follows:

$p$	$q$	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

The *and* operator sometimes is written as multiplication, or just  $pq$ . Most often, that is paired with using a bar for negation, so  $p \wedge \neg q$  is written  $p\bar{q}$ . This is common notation in electrical engineering. Occasionally the negation will

be written as  $q'$ . Programming languages may represent  $\wedge$  with operators or functions like `&&`, `and()`, `.AND.`, `and`.

The English word **or**, however, has quite a few different meanings. Sometimes we mean the logical *exclusive-or*, where one choice rules out the other, and sometimes we mean logical *or*, where both choices are possible.

In symbolic logic, the **disjunction** operator,  $\vee$ , is the latter type of *or*. The *disjunction* is true when either sub-clause is true:

$p$	$q$	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

In electrical engineering, *or* often is represented by  $+$ , so  $p \vee \neg q$  would be written  $p + \bar{q}$ . Programming languages may represent  $\vee$  with operators or functions like `||`, `or()`, `.OR.`, `or`.

The **exclusive-or** operator, which we will denote  $\oplus$ , is true whenever exactly one operand is true:

$p$	$q$	$p \oplus q$
1	1	0
1	0	1
0	1	1
0	0	0

Mathematically, the symbol for the *exclusive-or* operator is not particularly standardized. The symbol  $\underline{\vee}$  sometimes appears, as does the operator `xor`. We will expand on the notation  $\oplus$  once we discuss addition of binary numbers.

Note that  $\oplus$  often is not considered a core operator. We can write  $p \oplus q$  as  $(p \vee q) \neg (p \wedge q)$ .

Programming languages rarely provide  $\oplus$  logical operators, although they often provide *exclusive-or* operators on the binary representations of integers. More on those in the next chapter. But if you notice, this is the negation of equality. So programming languages express the logical *exclusive-or* by negating the equality of two logical expressions.

In symbolic logic, equality of two logical statements is **equivalence**. When expressed as an operator, equivalence takes the symbols  $\equiv$ ,  $\Leftrightarrow$ , or  $\leftrightarrow$ . The reason for the arrow forms will become clear soon.

$p$	$q$	$p \equiv q$
1	1	1
1	0	0
0	1	0
0	0	1

Again,  $\equiv$  is not a core operator.  $p \equiv q$  is the same statement as  $(p \wedge q) \vee (\neg p \wedge \neg q)$ .

Programming languages represent equality with `==`, `=`, `equal?`, and many other forms.

A logical statement that always is true is a **tautology**. So  $p \vee \neg p$  is a tautology:

The sky is periwinkle or the sky is not periwinkle.

Symbolically,

$$\models p \vee \neg p.$$

A statement that always is false is a **contradiction**. Here  $p \wedge \neg p$  is an example:

The sky is blue and the sky is not blue.

A statement that is neither always true nor always false is logically **contingent**. So  $pq$  is contingent on the values of  $p$  and  $q$ .

We use the symbol for tautology,  $\models$ , to make certain equivalence statements unambiguous. Because we defined  $\equiv$  as an operator above, the plain statement

$$p \vee q \equiv q \vee p$$

appears contingent on its values. By adding  $\models$ , we make it clear that we are asserting that the statement is always true. To state that  $p \vee q$  is the same as  $q \vee p$ , we say

$$\models p \vee q \equiv q \vee p.$$

## 4 Properties of logical operators

Three properties from arithmetic also hold for the core logical operators  $\wedge$  and  $\vee$ :

**commutative** In language, *and* and *or* are commutative, or  $p$  and  $q$  is the same as  $q$  and  $p$ . Symbolically,  $\models p \wedge q \equiv q \wedge p$  and  $\models p \vee q \equiv q \vee p$ .

**associative** Again, linguistically we don't use parenthesis. Run-on statements are just as true or false as well-structured ones, so we expect *and* and *or* to be associative. We could build a large truth table to verify this, but for now let us just state that  $\models (p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$  and  $\models (p \vee q) \vee r \equiv p \vee (q \vee r)$ .

**distributive** As with sets, both operations distribute over the other. So  $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$  and  $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ .

As a demonstration of using a truth table to show equivalence,

$p$	$q$	$r$	$q \vee r$	$p \wedge (q \vee r)$	$p \wedge q$	$p \wedge r$	$(p \wedge q) \vee (p \wedge r)$
1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1
1	0	1	1	1	0	1	1
1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0

Which of these hold for  $\oplus$  and  $\equiv$ ? Over which operations may each be distributed?

## 5 Truth tables and logical expressions

### 5.1 De Morgan's laws

As an example of truth tables and a demonstration of some very useful logical laws, we examine De Morgan's laws.

In arithmetic, we know that  $-(1+2) = -1 + -2$ . Logical operations are similar to arithmetic, but not *that* similar.

We could reason about the two rules, but they serve as succinct examples of truth tables.

- $\neg(p \vee q) \equiv \neg p \wedge \neg q$ :

$p$	$q$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$	$p \vee q$	$\neg(p \vee q)$
1	1	0	0	0	1	0
1	0	0	1	1	0	1
0	1	1	0	1	0	1
0	0	1	1	1	0	1

- $\neg(p \wedge q) \equiv \neg p \vee \neg q$ :

$p$	$q$	$\neg p$	$\neg q$	$\neg p \vee \neg q$	$p \wedge q$	$\neg(p \wedge q)$
1	1	0	0	0	1	0
1	0	0	1	0	1	0
0	1	1	0	0	1	0
0	0	1	1	1	0	1

## 5.2 Logical expressions from truth tables

Say we are provided with a truth table showing all possible values for an unknown logical statement  $f(p, q)$ . From that truth table, we can construct a logical statement equivalent to  $f(p, q)$ .

Take:

$p$	$q$	$f(p, q)$
1	1	0
1	0	1
0	1	0
0	0	1

To construct  $f$ , we can combine all the true outputs with  $\vee$ . The terms to be combined are the  $\wedge$  of all the variables. In each  $\wedge$  expression, each variable is negated to make its value true.

So in the above table, there are two true entries. One has  $p \equiv 1$  and  $q \equiv 0$ , while the other has  $p \equiv 0$  and  $q \equiv 1$ . The two corresponding  $\wedge$ -statements are  $p \wedge \neg q$  and  $\neg p \wedge q$ . So an equivalent statement is

$$\models f(p, q) \equiv (p \wedge \neg q) \vee (\neg p \wedge q).$$

Alternately, we can list the false entries and combine their negations. This provides another equivalent statement,

$$\models f(p, q) \equiv \neg((p \wedge q) \vee (\neg p \wedge \neg q)).$$

We can use De Morgan's laws to show these are the same:

$$\begin{aligned} \models \neg((p \wedge q) \vee (\neg p \wedge \neg q)) &\equiv \neg(p \wedge q) \wedge \neg(\neg p \wedge \neg q) \\ &\equiv (\neg p \vee \neg q) \wedge (p \vee q) \end{aligned}$$

Neither of these are the *simplest* possible expression, but each is equivalent to  $f(p, q)$  and is constructed by a straight-forward recipe.

To simplify this expression, we use the distributive property and the fact that  $\models p \vee \neg p$ ,

$$\begin{aligned} \models f(p, q) &\equiv (p \wedge \neg q) \vee (\neg p \wedge q) \\ &\equiv (p \vee \neg p) \wedge \neg q \\ &\equiv 1 \wedge \neg q \\ &\equiv \neg q. \end{aligned}$$

There is another method for simplifying expressions which we will not cover. If you are interested in a more visual method for simplifying expressions over a few

variables, look up Veitch charts or Karnaugh maps from electrical engineering. With those forms, you draw a 2-d truth table and cover the true values with boxes of area  $2^k$ . This mechanism is particularly useful when you have *don't care* values, places where the output value does not matter.

## 6 Conditionals

### 6.1 English and the operator $\rightarrow$

One (compound) operator we have not mentioned so far is the **conditional**, the symbolic form of an *if-then* rule. A statement like

If the sky is blue, then it is not raining.

is translated to

the sky is blue  $\rightarrow$  it is not raining,

or  $p \rightarrow q$  using only symbols. Here  $p$  is the **antecedent** and  $q$  is the **consequent**.

Many English statements are forms of conditionals. For example,

- It is not raining when the sky is blue.
- Rain does not fall from a blue sky.
- A blue sky implies no rain.
- A blue sky is sufficient for it not to be raining.
- Not raining is necessary for a blue sky.

Many forms:

- If  $p$ , then  $q$ .
- $p$  implies  $q$ .
- $p$  only if  $q$ .
- **$p$  is sufficient for  $q$ .**
- **$q$  is necessary for  $p$ .**
- $q$  if  $p$ .

The two in bold are very important in mathematics and science because they are very, very common and often read incorrectly. *Sufficient* follows the arrow in  $p \rightarrow q$ , and *necessary* works backward. We will see why when we examine the appropriate truth table.

Colloquially, we can refer to  $p$  as a premise or hypothesis and  $q$  as a conclusion. However, we will stop using those terms when we discuss logical deduction. Identifying  $p$  as a premise is useful for reasoning about  $p \rightarrow q$ , but it introduces

ambiguity when we consider  $\rightarrow$  as an operator. Just like with  $\models$  and  $\equiv$ , additional symbols provide the appropriate context.

## 6.2 Defining $p \rightarrow q$

The truth table defining the conditional  $\rightarrow$  is slightly surprising:

$p$	$q$	$p \rightarrow q$
1	1	1
1	0	0
0	1	1
0	0	1

The first and last lines are expected. When truth implies truth or falsity implies falsity, the statement as a whole is true.

The second line also is expected. A true premise implying a false conclusion renders the statement false.

Now comes the surprise. A false hypothesis implying a *true* conclusion renders the statement as a whole *true*! This is only surprising because we are looking at one line at a time. Consider the general case where the premise is false. If you start reasoning from a false hypothesis, what is the result? Anything. So any time  $p$  is false, the statement as a whole is true.

We can break the conditional operator into core operators by listing the single negated output and applying De Morgan's laws:

$$\begin{aligned} \models p \rightarrow q &\equiv \neg(p \wedge \neg q) \\ &\equiv \neg p \vee q. \end{aligned}$$

So  $p \rightarrow q$  is a true statement whenever the conclusion  $q$  is true or when we start from a false premise and  $\neq p$  is true.

## 6.3 Converse, inverse, and contrapositive

Considering a few rules from arithmetic, we see that  $\rightarrow$  is *not* commutative! The form  $q \rightarrow p$  is the **converse** of  $p \rightarrow q$ , but the two statements are not equivalent. Similarly, we cannot simply negate terms to obtain the negation, so  $\neg(p \rightarrow q)$  is not the same as the **inverse**  $\neg p \rightarrow \neg q$ . There is one other form here that *is* equivalent. The **contrapositive**  $\neg q \rightarrow \neg p$  is equivalent to  $p \rightarrow q$ .

$p$	$q$	$p \rightarrow q$	converse: $q \rightarrow p$	inverse: $\neg p \rightarrow \neg q$	contrapositive $\neg q \rightarrow \neg p$
1	1	1	1	1	1
1	0	0	1	1	1
0	1	1	0	0	0
0	0	1	1	1	1

## 6.4 If and only if, or $\leftrightarrow$

One more form of the conditional is important because it is so frequently used, the phrase *if and only if*.

$p$  if and only if  $q$  is a double conditional or **biconditional**. It means  $p \rightarrow q \wedge q \rightarrow p$  and is written symbolically as  $p \leftrightarrow q$ . In text, *if and only if* often is abbreviated as **iff**.

Looking at its truth table, we see that  $\leftrightarrow$  is the same as equivalence:

$p$	$q$	$p \rightarrow q$	$q \rightarrow p$	$p \leftrightarrow q$
1	1	1	1	1
1	0	0	1	0
0	1	1	0	0
0	0	1	1	1

Which symbol you use is a matter of preference and emphasis.

## 7 Quantifiers

We are interested in two quantifiers for logical statements: *for all* and *there exists*. These are **universal** and **existential** quantifiers, respectively. To demonstrate these, I'm going to switch to **predicate logic**. Here properties have parameters, and the quantifiers describe possible values for those parameters. There is a third common quantifier, the **uniqueness** quantifier, but we will not explore it.

Say Dexter has fleas (which he doesn't). So far, we would simply associate this statement with a single variable. But to examine quantifiers, we need a bit more.

Let  $P(p)$  be the property of having fleas. This  $P(\text{Dexter})$  is a (false) statement that Dexter has fleas. We can generalize this to state that  $\exists p \in D : P(p)$  where  $D$  is the set of dogs. If  $p$  is Dexter, then  $P(p)$  would be stating that Dexter has fleas, so there does exist such a  $p$ . The colon ( $:$ ) is read as "such that" or "satisfies" and sometimes is replaced by a vertical bar  $|$  or the backwards epsilon-ish symbol  $\ni$ .

But Dexter does not have fleas, so in reality  $\exists p \in D : \neg P(p)$ . Because Dexter is a dog, we know not *all* dogs have fleas. So  $\exists p \in D : \neg P(p)$  is the same as

$\neg\forall p \in D : P(p)$ .

Translating to and from English needs attention to detail. The word *always* does not always translate into  $\forall$ . Some translations:

$P(p)$ is always true.	$\forall p : P(p)$ .
$P(p)$ is almost always true.	$\exists p : \neg P(p)$ .
There always is some way for $P(p)$ to be true.	$\exists p : P(p)$ .
Sometimes $P(p)$ .	$\exists p : P(p)$ .

And here are the reasons why we care about formalizing quantifiers in this class: **Negation is tricky, and composing or nesting quantifiers is tricky.**

## 7.1 Negating quantifiers

Symbolically, two rules apply:

- $\neg(\forall p : P(p))$  is the same as  $\exists p : \neg P(p)$ , and
- $\neg(\exists p : P(p))$  is the same as  $\forall p : \neg P(p)$ .

Reading the first aloud,

Stating that not all  $p$  satisfy  $P(p)$  is the same as saying there exists some  $p$  that satisfies  $\neg P(p)$ .

And the second,

Stating that there does not exist a  $p$  such that  $P(p)$  is the same as saying that all  $p$  satisfy  $\neg P(p)$ .

## 7.2 Nesting quantifiers

One other item to note: Different quantifiers are not operators, hence you cannot assume they commute! Quantifiers **nest**. As an example, consider two statements:

All homework questions have been answered by at least one student.  
Some student has answered all homework problems.

Translating the first (true) statement into a symbolic form gives

$$\forall q \in \text{questions} \exists s \in \text{students} : s \text{ answered } q.$$

The second (false) statement becomes

$$\exists s \in \text{students} \forall q \in \text{questions} : s \text{ answered } q.$$

The symbolic versions appear similar. The only difference is the order of the quantifiers. But the statements obviously have very different meanings.

The text gives a nice visual interpretation in Section 3.5 using diagrams akin to Venn diagrams from set theory. Here we consider a more syntactic approach.

Given the English and logic statements:

All homework questions have been answered by at least one student.  
 $\forall q \in \text{questions } \exists s \in \text{students} : s \text{ answered } q.$

You can read the latter as the former, or as

For all questions, there is some student who has answered that question.

Which student is meant depends on which question is considered.

Now consider the other statement:

Some student has answered all homework problems.  
 $\exists s \in \text{students } \forall q \in \text{questions} : s \text{ answered } q.$

Here the questions answered depends on which student is considered. And this statement says that *one* student has answered *all* questions.

For a more formal example, you read  $\forall p \in \mathbb{J}^+ \exists q \in \mathbb{J}^+ : p < q$  as

For all positive integers  $p$  there exists a positive integer  $q$  such that  $p > q$ .

The particular  $q$  depends on the  $p$ . There is not be a single  $q$  that works for all positive integers  $p$ . This statement is true over the integers.

Swapping the quantified statements produces a false statement. The translation of  $\exists q \in \mathbb{J}^+ \forall p \in \mathbb{J}^+ : p < q$  is

There exists a positive integer  $q$  such that for all positive integers  $p$ ,  $p < q$ .

There is no such integer, as  $q = q$  and thus  $q \not< q$ .

So we cannot simply swap quantifiers.

### 7.3 Combining nesting with negation

How do we negate the following true and equivalent English and logic statements?

All homework questions have been answered by at least one person.  
 $\forall q \in \text{questions } \exists s \in \text{students} : s \text{ answered } q.$

Consider working symbolically with the rules we already established:

$$\begin{aligned} & \neg(\forall q \in \text{questions } \exists s \in \text{students} : s \text{ answered } q) \\ & \equiv \exists q \in \text{questions } \neg(\exists s \in \text{students} : s \text{ answered } q) \\ & \equiv \exists q \in \text{questions } \forall s \in \text{students} : \neg(s \text{ answered } q) \\ & \equiv \exists q \in \text{questions } \forall s \in \text{students} : s \text{ has not answered } q. \end{aligned}$$

So the negation is

There exists a question such that for all students the student has not answered the question.

Equivalently,

There is some question where no student has answered that question.

Is this the same as just sticking a *not* in front of the original sentence?

Not all homework questions have been answered by at least one person.

In this case, yes. But the other statement is not as simple.

Now consider the other statement:

Some student has answered all homework problems.  
 $\exists s \in \text{students } \forall q \in \text{questions} : s \text{ answered } q.$

Does the statement

Not some student has answered all homework problems.

mean anything? Not really. So how can we negate this statement correctly?

Work with the symbolic form. Then

$$\begin{aligned} & \models \neg(\exists s \in \text{students } \forall q \in \text{questions} : s \text{ answered } q) \\ & \equiv \forall s \in \text{students } \neg(\forall q \in \text{questions} : s \text{ answered } q) \\ & \equiv \forall s \in \text{students } \exists q \in \text{questions} : \neg(s \text{ answered } q) \\ & \equiv \forall s \in \text{students } \exists q \in \text{questions} : s \text{ has not answered } q. \end{aligned}$$

So a correct negation is

For all students, there is some question that student did not answer.

## 8 Logical deduction: Delayed until after the test

Now that we have *if-then* statements, we can speak more about *logical deduction*. And the simplest level, we can chain  $\rightarrow$  operators to show deduction. But then

we end up with statements like  $((p \rightarrow q) \wedge (q \rightarrow r) \wedge \neg r) \rightarrow ((p \rightarrow r) \wedge \neg r) \rightarrow p$ . These become unreadable quickly.

Instead, we introduce new symbols. The above could be written as  $p \rightarrow q, q \rightarrow r, \neg r \vdash p \rightarrow r, \neg r \vdash p$ . Taking up more room but being more clear, we can write the **logical argument** or **logical deduction** as

$$\begin{array}{c} p \rightarrow q \\ q \rightarrow r \\ \neg r \\ \hline \vdash p \rightarrow r \\ \neg r \\ \hline \vdash \neg p \end{array}$$

The general form in one line: Premise 1, premise 2  $\vdash$  conclusion. In table form,

$$\begin{array}{c} \text{Premise 1} \\ \text{Premise 2} \\ \hline \vdash \text{Conclusion.} \end{array}$$

Both are read as *assuming premise one and premise two, infer the conclusion* or *premise one and premise two entail the conclusion*.

The  $\vdash$  symbol is *syntactic sugar* meant to place emphasis where important.  $p \vdash q$  is the same as  $\vdash p \rightarrow q$  or just  $p \rightarrow q$ , but the former implies a deduction while the latter appears to be just a statement.

Sometimes three dots,  $\therefore$ , is used instead of  $\vdash$ . And sometimes (as in the text) neither appears in the tabular form. However, symbolic logic is about removing ambiguity that comes from language. Using the symbol helps disambiguate valid logical arguments from examples of invalid arguments or fallacies.

When reasoning about reasoning itself, the set of premises in a rule often are represented by  $\Gamma$ , so  $\Gamma \vdash q$  is a rule with a *set* of premises  $\Gamma$  leading to a single result  $q$ .

Some forms or structures of logical arguments have classical names. These names came long before the symbolic form.

**Modus ponens**  $p \rightarrow q, p \vdash q$ , or

$$\begin{array}{c} p \rightarrow q \\ p \\ \hline \vdash q \end{array}$$

**Modus tollens**  $p \rightarrow q, \neg q \vdash \neg p$ , or

$$\begin{array}{c} p \rightarrow q \\ \neg q \\ \hline \vdash \neg p \end{array}$$

**Disjunctive syllogism**  $p \vee q, \neg p \vdash q$ , or

$$\frac{p \vee q}{\neg p} \vdash q$$

**Hypothetical syllogism** (also called **transitivity**)  $p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$ , or

$$\frac{p \rightarrow q}{q \rightarrow r} \vdash p \rightarrow r$$

We also can write De Morgan's laws as laws of deduction, for example  $\neg(p \vee q) \vdash \neg p \wedge \neg q$ , or

$$\frac{\neg(p \vee q)}{\vdash \neg p \wedge \neg q} .$$

Much of the reason why we care about the *correct* rules of deduction is to highlight incorrect rules, or **logical fallacies**. A list of a few common fallacies follows. Note that we include a symbol after the line in the tabular form; the negation of implication,  $\not\vdash$ , lets you know we are talking about fallacies. The text does not use any symbol, so you may end up seeing these fallacies as valid.

**Fallacy of the converse** Given  $p \rightarrow q$  and  $q$ , we cannot deduce  $p$ . Symbolically,  $p \rightarrow q, q \not\vdash p$  or

$$\frac{p \rightarrow q}{q} \not\vdash p .$$

**Fallacy of the inverse** Given  $p \rightarrow q$  and  $\neg p$ , we cannot deduce  $\neg q$ . Symbolically,  $p \rightarrow q, \neg p \not\vdash \neg q$  or

$$\frac{p \rightarrow q}{\neg p} \not\vdash \neg q .$$

**Fallacy of the alternative disjunct** Given  $p \vee q$  and  $p$ , we cannot deduce  $\neg q$ . Symbolically,  $p \vee q, p \not\vdash \neg q$  or

$$\frac{p \vee q}{p} \not\vdash \neg q .$$

With the *exclusive-or* operator  $\oplus$ , we can conclude that only one disjunct is chosen. But the *or* operator  $\vee$  allows both to occur at once.

## 9 Homework

**Practice is absolutely critical in this class.**

Groups are fine, turn in your own work. Homework is due in or before class on Mondays.

- Section 3.1
  - Problems 1-5
  - Problems 40, 42, 44
  - Problems 49-54
- Section 3.2
  - Problems 15-18
  - Problems 37-40
  - Problems 53-55
  - Problems 61, 62
- Section 3.3
  - Problems 1-5
  - Problems 13, 15, 20
  - Problems 35-38
  - Problems 58, 60
  - Problems 67, 68
  - Problems 74, 75
- Section 3.4
  - Problems 1, 3, 6
  - Problem 51, 57, 58
- Section 3.1 again
  - Problems 55, 56
  - Problems 60-64
  - Problem 75
  - Problem 76. Hint: Quantifiers do not necessarily exclude each other.
- Negate the following, and decide if the statements are true or false.
  - There is a number  $p$  for all numbers  $q$  such that the difference between  $p$  and  $q$  is 2.

- For all sets  $A$ , for all sets  $B$ , there is a set  $C$  such that  $A \cap B = C$  and  $C$  is not  $\emptyset$ .
- Derive a logic expression from the following truth table. Attempt to simplify it remembering the distributive property, De Morgan's laws, and that  $\models z \vee \neg z \equiv 1$  and  $\models z \wedge \neg z \equiv 0$ .

$p$	$q$	$r$	$f(p, q, r)$
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

- 
- Section 3.6: **Delayed until after the test week.**

- Problems 3, 6
- Problems 17, 19, 21
- Problems 47, 49

---

Note that you *may* email homework. However, I don't use Microsoft<sup>TM</sup> products (*e.g.* Word), and software packages are notoriously finicky about translating mathematics.

If you're typing it (which I advise just for practice in whatever tools you use), you likely want to turn in a printout. If you do want to email your submission, please produce a PDF or PostScript document.