Intro to Mathematical Reasoning (Math 300)

Supplement 2. **Analyzing complex predicates and pushing negations through a predicate** [1]

Important note: *There is a lot of information in this supplement and many examples. You are not expected to understand all the details the first time you read it. The material in this supplement will be important for this entire course (and in your future courses) so you should expect to refer back to it throughout the course.*

On first reading, you want to understand as much as possible and to mark those things that you don't understand. Try to formulate specific questions to ask your instructor about the material you don't understand.

In this supplement, you will learn some specific technical skills for working with complicated mathematical statements and predicates. One such skill is constructing a "useful denial" of a statement or predicate.

In sections 1.1 and 1.3 of the text, you were introduced to the *negation* of a statement or predicate and a *denial* of a statement or predicate. As is stated there, every statement has exactly one negation: the negation of a sentence A is the sentence "It is not the case that A". A statement that is equivalent to the negation of A is called a *denial* of A. A statement has many denials. The book talks about the problem of transforming the negation of A into a "useful denial" of A. This concept of a useful denial is important but vague. One of our goals in this handout is to clarify this concept.

Let's start with a simple example. Consider the predicate:

Predicate $P(x)$: $x = 2$ or $x \geq 4$,

which is a predicate in the variable x. The negation of this predicate is:

Predicate $Q(x)$: It is not the case that $x = 2$ or $x \geq 4$.

In the book the author explains that the negation of the "or" of two statements is equivalent the "and" of their negations. In this case, $Q(x)$ is equivalent to:

Predicate $Q'(x)$: $x \neq 2$ and $x < 4$.

When we say that these predicates are *equivalent*, we mean that the set of values for x that make $Q(x)$ true is exactly the same as the set of values that make $Q'(x)$ true.

The process of modifying a statement or predicate so as to eliminate negations is called *pushing negations through* the statement or predicate. Let's look at a more complicated mathematical sentence (which we saw already in Supplement 1):

Statement S: For all integers n, if n is prime and $n - 1$ is divisible by 4, then there are integers a and b such that $n = a^2 + b^2$.

[1] Version 9/6/04.

It's negation is:

> **Statement** T: It is not the case that for all integers n, if n is prime and $n-1$ is divisible by 4, then there are integers a and b such that $n = a^2 + b^2$.

Here is the equivalent statement obtained by pushing negations through T:

> **Statement** T': There is a prime integer n such that $n-1$ is divisible by 4, and such that for all integers a and b, $n \neq a^2 + b^2$.

For an experienced mathematician, constructing T' from T is nearly automatic, requiring little conscious thought. In contrast, students beginning in math 300 are rarely able to do it correctly. One of the purposes of this supplement is to give a systematic method for pushing negations through sentences. The method is a bit tedious at first, but with practice you will learn to do it more efficiently.

Pushing negations through a sentence is done in stages. The first and most important stage consists of *parsing* the sentence. Roughly, this means figuring out how the sentence is put together from smaller pieces. Once we have parsed a sentence, pushing negations through will be fairly straightforward. Parsing is important for more than simplifying negated sentences. It is fundamental part of understanding, applying, and proving a complicated sentence. So the main thing is to learn what parsing is, and how to do it.

Before discussing parsing we need to review the notions of statement and predicate and carefully discuss the notions of free variables and bound variables.

1 Predicates, Statements, free and bound variables

As introduced in Section 1.1 of the text, a mathematical proposition (or statement) is a sentence that, in principle, can be said to be true or false (even if we don't know which it is). The concept of a *predicate* is introduced in Section 1.3. A predicate is a sentence that may have one or more *free variables*, that becomes a statement if a value is selected for each of the free variables. Each free variable has a set of allowed values, which is usually known from context. Note that the word "value" here does not necessarily mean "number". Depending on the context, the value of a variable may be a number, a vector, a matrix, a set, a function, or any other mathematical object.

Statements can be viewed as a predicates that have no free variables. Thus the things we say here about predicates also apply to statements.

Here are two examples of predicates:

Predicate A**:** For all real numbers x that belong to the set S, $x \leq 10$.

Predicate B**:** $\sum_{i=0}^{n} y^i = \frac{y^{n+1}-1}{y-1}$

Variables in a predicate that are not free are called *unfree* or *bound*. In supplement 4, we will refer to them as *dummy variables*. It is essential to be able to distinguish free variables from bound variables in sentences. Given a sentence written in mathematical English, there is no simple rule that works all the time but there are three basic guidelines that together will enable you to distinguish the two types of variables in a predicate.

1. Variables that appear with a universal or existential quantifier are bound, but are not the only bound variables. Other examples include:

   (a) The index of summation in a sum.

   (b) The variable of integration in a definite integral

   (c) A dummy variable in the description of a set. (This will be explained later.)

2. If you take a free variable that appears in a sentence and choose an allowed value for it and replace every occurence of that variable by the chosen value then the sentence must make sense. *Important: The sentence* need not be true*, it need only make sense.* If you do this for all free variables then the result will be a mathematical statement. If you substitute a value for a bound variable the sentence will not make sense, or at least will be very strange.

3. If a variable is bound, then it is possible to rewrite the sentence so that it has exactly the same meaning but the variable does not appear in the sentence. This can not be done with a free variable.

In predicate A above there are two variables x and S. The variable x represents a number while the variable S represents a set.

Let's use each of the three guidelines to classify the variables of A as bound or free. Using the first guideline: Here the variable x appears with the universal quantifier, and so it is bound. The variable S is free.

Now use the second guideline. Suppose we choose a specific set, the set of all even numbers, and substitute it for S in predicate A:

> For all real numbers x that belong to the set of even numbers, $x \leq 10$.

This sentence makes sense, though it is false statement. This indicates that S is a free variable.

Suppose we substitute the number 3 for x in A:

> For all real numbers 3 that belong to the set S, $3 \leq 10$.

This sentence does not make sense as an English sentence and indicates that the variable x is bound.

Finally, considering the third guideline, we check that we can rewrite sentence A so that x does not appear:

> Every number belonging to the set S is less than or equal to 10.

There is no x in sight. Since x can be eliminated from the sentence without changing the meaning, it is a bound variable. On the other hand, you can't eliminate S without changing the meaning of the sentence.

Let's consider predicate B. There are three variables i,n and y. Here n represents an integer, i represents an integer between 0 and n. Without additional information, it is not entirely clear what kind of object y, but from context it seems that y represents a number.

Using the first guideline, i is an index of summation, which means it changes for different terms of the sum, and therefore it is a bound variable. The free variables are y and n.

Using the second guideline, if we substitute 6 for n and 3.5 for y we get the statement:

$\sum_{i=0}^{6}(3.5)^i = ((3.5)^7 - 1)/(3.5 - 1)$,

which makes sense, and happens to be true.

If we substitute 3 for i in B we get nonsense:

$\sum_{3=0}^{n} y^3 = \frac{y^n - 1}{y - 1}$,

Finally, considering the third guideline, we can reexpress the sentence without i as:

> The sum of the terms obtained by raising y to each integer power from 0 up to n is equal to $(y^{n+1} - 1)/(y - 1)$.

This is long, but is equivalent to B. (It also shows why summation notation is useful for expressing things efficiently.)

A final remark for this section: It is customary to name predicates by a single capital letter and to follow the letter by the free variable (or list of free variables, if there is more than one) in parentheses. For example, predicate B above might be written as $B(y, n)$.

2 Building up a mathematical predicate or statement

There are seven basic operations that can be used to build up complex mathematical predicates from simpler ones.

- A predicate can be negated.
- Two mathematical predicates can be combined together using one of the four *logical connectives*:
  - "and" ($\wedge$)
  - "or" ($\vee$)
  - "if–then" or "implies" ($\Longrightarrow$)
  - "if and only if" ($\Longleftrightarrow$)
- We can *quantify over a free variable* using a quantifier to make one of the free variables into a bound variable.
  - Existential quantification using "there exists" ($\exists$)
  - Universal quantification using "for all" ($\forall$) For example, the predicate "$x = y^2$" has free variables x and y. We can quantify over x to get either of the sentences "There exists x such that $x = y^2$" or "For all x, $x = y^2$".

A predicate that contains no logical connectives, quantifiers, or negations is said to be an *primitive predicate*.

When two predicates are combined using one of the four logical connectives, the set of free variables of the new predicate is the union of the sets of free variables of the old predicates. Thus

the predicate "$n \geq a$" has free variables n and a and the predicate "$n \leq j + k$ has free variables n, j, k so the predicate "$n \geq a$ and $n \leq j + k$" has free variables n, a, j, k.

Mathematical predicates that are built up properly from primitive predicates using these seven ways of building sentences are said to be *well formed* mathematical predicates. A well formed sentence need not be true. For example, "For all real numbers x, $x < 0$ or $x = 1$" is well formed but not true. The predicate "$x < y$ and $x > y$" is well formed even though it is impossible to find two real numbers to substitute for x and y to make it true.

When a sentence is not well formed it is not clear what the sentence even means. For example "For all x or y, $x + y$ is even" is *not well formed.*

Here are two important rules for constructing well formed sentences.

- You can not bind a variable that is already bound. For example, the sentence "For all x, there exists x such that $x^2 \geq x$" is nonsense because the variable x is bound in "There exists x such that $x^2 \geq x$" so we can't bind it again with "For all x".

  Note that the sentence "For all real numbers x, $x^2 \geq x$ and there exists x such that $x^2 = x$" is a well formed statement. It is obtained by combining the two statements "For all real numbers x, $x^2 \geq x$" and "there exists x such that $x^2 = x$" with the logical connective "and".

- Normally, when we have two well formed predicates, and we combine them with one of the four logical connectives, the result is well formed. However, there is one important exception. If P and Q are predicates where x is free in one and bound in the other, then P and Q should not be combined. For example, if P is "$x \leq y^2$" and Q is "There exists a real number y such that $x \leq y^3$" then the sentence "$x \leq y^2$ and there exists a real number y such that $x \leq y^2$." is *not well formed.* This is because *a variable should never appear in the same sentence as both a bound and a free variable.*

  If we want to combine these sentences we should first *rename the bound variable.* When dealing with mathematical sentences, you are always permitted to change the name of any bound variable (but you are not generally permitted to change the name of a free variable.) So sentence Q can be rewritten as the sentence Q' ="There exists a real number z such that $x \leq z^2$" and then "P and Q'" is well formed.

Let us look at an example of a complicated sentence to see how it is put together from primitive predicates. Let us use statement S from above:

> **Statement** S: For all integers n, if n is prime and $n - 1$ is divisible by 4, then there are integers a and b such that $n = a^2 + b^2$.

This sentence is composed from three primitive predicates.

$C(n) =$ "n is prime",

$D(n) =$ "$n - 1$ is divisible by 4",

$E(n, a, b) =$ "$n = a^2 + b^2$".

Each of these primitive predicates is given a capital letter name for reference, and the letter is followed by the list of free variables in parentheses.

We can abbreviate our sentence as:

$\forall$ integers n, if $C(n)$ and $D(n)$ then $\exists$ integers a, b such that $E(n, a, b)$.

This sentence is built up from the primitive predicates by the following steps:

1. Combine $C(n)$, $D(n)$ with the connective "and" to get the predicate $F(n) = C(n) \wedge D(n)$.
2. Apply existential quantification over the variables a and b to the predicate $E(n, a, b)$ to obtain the predicate $G(n) =$"$\exists$ integers a, b such that $E(n, a, b)$". (Note that a, b are not free variables in G).
3. Combine $F(n), G(n)$ with the "if–then" connective to get the predicate $H(n)$ which is $F(n) \Longrightarrow G(n)$.
4. Apply universal quantification over the variable n to $H(n)$ to obtain the predicate (statement) $S =$"$\forall$ integers $n, H(n)$.

Thus S was built up by a list of predicates:

1. $C(n)$,
2. $D(n)$,
3. $E(n, a, b)$,
4. $F(n)$ is "$C(n) \wedge D(n)$",
5. $G(n)$ is "$\exists$ integers a, b such that $E(n, a, b)$",
6. $H(n)$ is "$F(n) \Longrightarrow G(n)$",
7. S is "$\forall$ integers $n, H(n)$".

where the first three predicates in the sequence are primitive predicates and each of the other predicates in the list is obtained from one or two of the previous predicates by one of the seven operations on predicates. The predicates $F(n)$, $G(n)$ and $H(n)$ are the *intermediate predicates* in the construction and S is the *final predicate*.

We call this *predicate list analysis* of the predicate S. This list provides a detailed picture of how S is built up from primitive predicates. The final expression used to construct the predicate is called the *top level structure* of the predicate. For example, the top level structure of the predicate S is "$\forall$ integers n, $H(n)$", while the top level structure of the predicate $H(n)$ is $F(n) \Longrightarrow G(n)$.

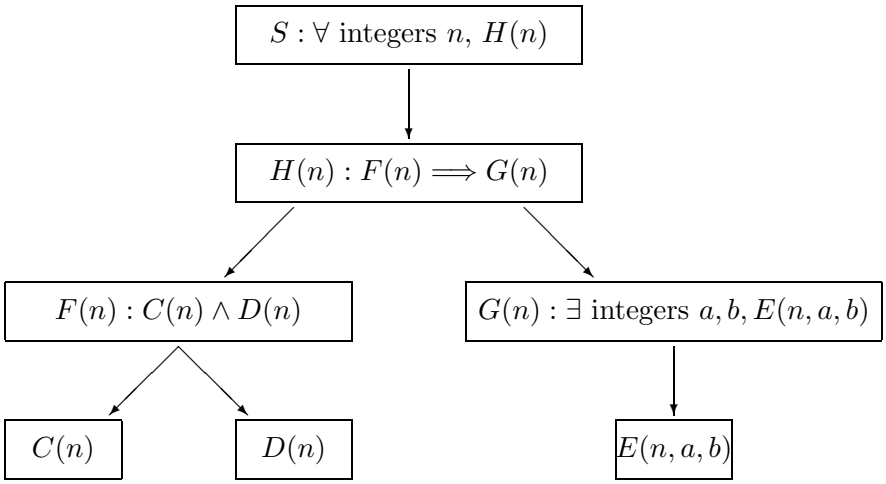
Figure 1 provides a diagram of this process called a *parse tree*. Each predicate appearing in the tree is called a *node* of the tree. The top node is the sentence being analyzed and the bottom nodes are the primitive predicates. Each node not at the bottom has one or two branches to the one or two predicates from which it is built.

It is easy to see that a parse tree for a predicate can be used to build a predicate list representation, and vice versa. Constructing one or the other of these is called *parsing* the predicate.

We can also represent most of the information in the parse tree using parentheses:

$$\forall n((C(n) \wedge D(n)) \Longrightarrow \exists a, b E(n, a, b))$$

Figure 1: Parse tree for the statement S.



The purpose of the parentheses here is similar to the purpose of parentheses in arithmetic expressions: to specify the order in which the parts are combined. By working in from the outer parentheses we can reduce the sentence into its smaller subparts. The outermost left and right parenthesis enclose the predicate we called $H(n)$ above. After removing the outer parentheses, we get:

$$(C(n) \wedge D(n)) \Longrightarrow \exists a, b E(n, a, b),$$

the first part $(C(n) \wedge D(n)$ is what we called $F(n)$ and the second part $(\exists a, b, E(n, a, b))$ is what we called $G(n)$.

3 Negating Sentences

Once we have parsed a sentence correctly, pushing negations through is straightforward.

Corresponding to each of the seven ways of building up predicates is a negation rule. Some of these rules are given in the text, but let's repeat them here. For convenience we write these for predicates with one free variable, but they extend to other predicates. For any two predicates $A(x)$ and $B(x)$:

$$\begin{array}{rll}
\sim (A(x) \wedge B(x)) & \text{is equivalent to} & (\sim A(x)) \vee (\sim B(x)) \\
\sim (A(x) \vee B(x)) & \text{is equivalent to} & (\sim A(x)) \wedge (\sim B(x)) \\
\sim (A(x) \Longrightarrow B(x)) & \text{is equivalent to} & A(x) \wedge \sim B(x) \\
\sim (A(x) \Longleftrightarrow B(x)) & \text{is equivalent to} & (A(x) \wedge \sim B(x)) \vee (B(x) \wedge \sim A(x)) \\
\sim (\sim A(x)) & \text{is equivalent to} & A(x)
\end{array}$$

$$\sim (\exists x, A(x)) \quad \text{is equivalent to} \quad \forall x, \sim A(x)$$
$$\sim (\forall x, A(x)) \quad \text{is equivalent to} \quad \exists x, \sim A(x)$$

To push negations through a statement means to apply these rules repeatedly to a sentence to get a sentence where the only negations within the parse tree are applied to primitive predicates. This can be done using the parse tree by working down from the top level (the final predicate). Let's do this on the example sentence S whose parse tree was worked out earlier in Figure 1.

The top level structure of S is "$\forall$ integers n, $H(n)$". When we negate S, we can apply the negation rule for "for all" statements above and say:

$\sim S$ is equivalent to "$\exists$ integer n, such that $\sim H(n)$."

Next we replace H by its top level structure:

$\sim S$ is equivalent to "$\exists$ integer n such that $\sim (F(n) \Longrightarrow G(n))$".

Then we apply the negation rule for "if–then" predicates:

$\sim S$ is equivalent to "$\exists$ integer n such that $F(n) \wedge \sim G(n)$."

Notice that $F(n)$ is not negated so we don't have to do anything more with it. $G(n)$ is negated so we expand $G(n)$ using its top level structure:

$\sim S$ is equivalent to "($\exists$ integer n such that $F(n)) \wedge \sim$ ($\exists$ integers a, b such that $E(n, a, b)$)".

Applying the rule for negating "there exists" we get:

$\sim S$ is equivalent to "($\exists$ integer n such that $F(n)) \wedge \forall$ integers $a, b, \sim E(n, a, b)$."

At this point, the only negated predicate is $E(n, a, b)$ which is primitive. So we are finished with this part. Next we rewrite this in English:

> There exists an integer n such that n is prime and 4 is a divisor of $n - 1$ and such that for all integers a, b it is not the case that $n = a^2 + b^2$.

Here we can go a little further. For the primitive predicate $n = a^2 + b^2$ we have standard notation for the negation, namely $n \neq a^2 + b^2$.

> There exists an integer n such that n is prime and 4 is a divisor of $n - 1$ and such that for all integers a, b $n \neq a^2 + b^2$.

We can rewrite this as:

> There exists a prime integer n such that 4 is a divisor of $n - 1$ and such that for all integers a, b $n \neq a^2 + b^2$.

Now, done in this detail, this process is very tedious. When working out your first examples, it is important to go through the process to make sure that you follow what's going on. As you gain experience, you will probably find that you, like most experienced mathematicians, can do this in your head.

4 Ambiguities in parsing

Note: The remainder of this handout deals with some important but subtle issues about parsing sentences. The first time you read the handout, you should skim or skip this part.

The process of parsing a sentence, either by constructing the parse tree or the predicate list analysis is usually straightforward. However, there are circumstances which make parsing unclear.

As a first example, consider the predicate

$x \geq 10$ and $x \leq 20$ or $x^2 + 3x \leq 30$.

This predicate has three primitive components, which we'll call $A(x)$, $B(x)$ and $C(x)$. Symbolically we rewrite $A(x) \wedge B(x) \vee C(x)$. Now the problem is: does this mean $(A(x) \wedge B(x)) \vee C(x)$ or $A(x) \wedge (B(x) \vee C(x))$? Here the sentence has more than one possible meaning. This situation is referred to as an *ambiguity*.

The problem of dealing with ambiguities is a problem for *both the writer and the reader*. As a reader, it is your responsibility to figure out what the writer meant. When you encounter a sentence that seems ambiguous, you need to look for clues that will allow you to *resolve the ambiguity*, which means to figure out the one meaning the writer had in mind. As a writer, it is your responsibility to make it as easy for the reader as possible, and part of this responsibility means *anticipating* and *avoiding* any possible ambiguities. In this section we will discuss some of the ways possible ambiguities arise, and how to deal with them as both reader and writer.

You are probably very familiar with the problem of ambiguities in arithmetic, where the expression $a + b \times c$ could mean $(a + b) \times c$ or $a \times (b + c)$. In arithmetic, we place the parentheses in to make things clear. Also, there are standard rules of precedence which say, for example, that if there are no parentheses then the multiplication is done before the addition so that $a + b \times c$ means $a + (b \times c)$.

Dealing with ambiguities in mathematical statements is less clear cut then dealing with ambiguities in arithmetic because arithmetic is written symbolically, while mathematical statements are usually written in English. In English, we don't normally use parentheses in our sentences to eliminate ambiguities, so we need other ways. However, at this point in your writing, the most important thing is that your writing be understood. If using parentheses in your sentences makes the meaning clearer, then do it!

Here are ways that mathematicians use in their writing to eliminate ambiguities. The first is by the careful use of commas. In the sentence "$x \geq 10$ and $x \leq 20$, or $x^2 + 3x \leq 30$" the comma separates the sentence into two parts "$x \geq 10$ and $x \leq 20$" and "$x^2 + 3x \leq 30$. This indicates that the "or" operation is done after the "and". Similarly, if we write "$x \geq 10$, and $x \leq 20$ or $x^2 + 3x \leq 30$" this normally indicates that the "and" is done after the "or".

Another option when you intend the "and" to be done last is to write "and also" in place of "and". Thus in the sentence "$x \geq 10$ and also $x \leq 20$ or $x^2 + 3x \leq 30$" the word also serves to set "$x \geq 10$" apart from the rest. Using both the comma and the word "also" makes it even clearer: "$x \geq 10$, and also $x \leq 20$ or $x^2 + 3x \leq 30$"

If you want the "or" to be done last, then you can't write "or also". You can emphasize that the "and" should be done first by using the word "both" as in: "Both $x \geq 10$ and $x \leq 20$, or $x^2 + 3x \leq 30$". In this case, it is even clearer to rearrange the sentence and write: "$x^2 + 3x \leq 30$, or both $x \geq 10$ and $x \leq 20$."

Sometimes there seems to be a possible ambiguity, but a bit of thought reveals that one of the interpretations is impossible. Here's an example:

> For all integers n that belong to the set S, n is even and there exists an integer k such that $k|n^2$.

Let us write $A(n)$ for the predicate "n is even", and $B(n,k)$ for the predicate "$k|n^2$". There are two possible interpretations:

> ($\forall$ integers n that belong to the set S, $A(n)$) and ($\exists$ integer k, $B(n,k)$)

or

> $\forall$ integers n that belong to the set S, ($A(n)$ and $\exists$ integer k, $B(n,k)$)

Here the second is the only one that makes sense. In the first (and incorrect) interpretation, the first part is a predicate with free variable S and bound variable n while the second is a predicate in the free variables n and k. However, the rules for well formed predicates prohibits combining two sentences where one has n as a free variable and the other has n as a bound variable.

Let us contrast this with another very similar example. The only change is that the second quantifier is over n rather than k.

> For all integers n that belong to the set S, n is even and there exists an integer n such that $k|n^2$.

Again there are two possible interpretations:

> ($\forall$ integers $n \in S$, $A(n)$) and ($\exists$ integer n, $B(n,k)$).

or

> $\forall$ integers $n \in S$, ($A(n)$ and $\exists$ integer n, $B(n,k)$).

Here it is the second interpretation that can be eliminated. In that sentence, the predicate $A(n)$ has n as free variable and the predicate "$\exists$ integer n $B(n,k)$" has n as bound, so they can't be combined with "and". So the only remaining interpretation is the first one.

Next we discuss some situations which are not exactly ambiguous, but where the flexibility of English writing might cause confusion. As was explained when we discussed free and bound variables, when we have a bound variable, it is possible to rewrite the sentence so that the variable disappears. To parse a sentence, we want to rewrite the sentence so that each quantifier has an associated variable. Also, in English, we might write something like "$ab \leq (a+b)^2/4$ for all real numbers a and b" where the quantifier "for all" comes after the predicate being quantified. But before analyzing the statement, it is less confusing to rewrite the statement as "for all real numbers a and b, $ab \leq (a+b)^2/4$". This is not absolutely necessary, but with complicated sentences there is less chance for confusion if quantifiers come before the predicate to which they refer.

Here's an example:

> If there are no integers greater than 1 that divide both a and b then $ax + by = 1$ for some integers x and y.

The part between the "if" and "then" says "there are no integers greater than 1 that divide a and b". The phrase "there are no integers" is the negation of "there exists an integer". Also there is no explicit variable being quantified. We can rewrite this part as:

"There is no integer d such that $d > 1$ and $d|a$ and $d|b$."

The part of the sentence after "then" has the quantifier at the end so we rewrite it as:

"There exist integers x and y such that $ax + by = 1$.

Thus we arrive at the following restatement of the original sentence.

If there is no integer d such that $d > 1$ and $d|a$ and $d|b$, then there exist integers x and y such that $ax + by = 1$.

Note that this sentence is a predicate with two free variables a and b and three bound variables d, x and y.

Rewriting the sentence in this way is not absolutely necessary, but for many students it helps avoid confusion.

5 Implicit quantifiers

Another situation where written English allows more freedom than precise mathematical writing is in the use of *implicit universal quantifiers*. This was discussed in supplement 1. For example, the sentence

Predicate U: If n is a prime then n is odd.

is technically not a statement but a predicate with free variable n. However, this is usually interpreted as though there were a universal quantifier for n in front.

Statement V: For all n, if n is a prime then n is odd.

(Of course, this is a false statement.) You were warned earlier that leaving out the quantifier can cause all kinds of trouble. One specific place it can cause trouble is when you negate a sentence. If you negate the first sentence and push negations through you get:

Predicate W: n is a prime and n is not odd,

which is a predicate in n. It is equivalent to the negation of predicate V but is not equivalent to the negation of statement V. If you negate V and push negations through you get:

Statement X: There exists an n such that n is a prime and n is not odd.

This is one of the main reasons it is much safer not to write U when what you mean is V. If you need to negate the sentence then you may get it wrong.

6 Analyzing primitive predicates

In what we presented above, the primitive predicates serve as the building blocks for our sentences. But in fact, it is often possible to go further and to analyze the primitive predicates. The reason for this is that while a primitive predicate contains no logical connectives or quantifiers, it may be an abbreviation for a more complex predicate that is not primitive. This is best understood by considering an example. Suppose we return to the sentence "For all n, if n is prime then n is odd or $n = 2$". We can apply the above method to construct a predicate list analysis and a parse tree. The primitive predicates are "n is prime", "n is odd" and "$n = 2$". While all three are primitive predicates, the first two are actually abbreviations for more complicated predicates. For example, "n is prime" is an abbreviation for:

> $n > 1$ and for all positive integers d if d is a divisor of n then $d = 1$ or $d = n$.

This predicate is made up of four primitive predicates "$n > 1$", "d is a divisor of n", "$d = 1$" and "$d = n$".

When we replace "n is prime" by this more complex predicate in the original sentence we get more detail about the meaning of the sentence. We can break this down further by using the definition of divisor to replace the predicate "d is a divisor of n" as "there exists an integer k such that $n = kd$"

Thus "n is prime" is equivalent to:

> $n > 1$ and for all positive integers d if there exists an integer k such that $n = kd$ then $d = 1$ or $d = n$.

Notice that primitive predicates are "$n > 1$", "$n = kd$", "$d = 1$" and "$d = n$" which are all very simple.

A similar thing can be done with the predicate "n is odd". Using the definition of odd, this means "There exists an integer k such that $n = 2k+1$" whose only primitive predicate "$n = 2k+1$" is very simple.

Important: This section is *not* saying that you *should* replace the predicate "n is prime" by the longer predicate with simpler atoms. Rather it is pointing out (1) that primitive predicates, as we have defined them, can be abbreviations for more complicated predicates, and (2) it may sometimes be useful to expand a primitive predicate using its definition. We will see situations where it is useful to expand primitive predicates and you should keep this in the back of your mind.

In general, however, expanding a primitive predicate usually makes the sentence longer and harder to understand. You *should not* expand primitive predicates *unless there is a reason to do so.*