Mathematics for Computer Science
用于计算机科学的数学知识

revised Wednesday 8th September, 2010, 00:40

2010年9月8日星期三，00:40修订

Eric Lehman

Google Inc.

谷歌有限公司

F Thomson Leighton

Department of Mathematics and CSAIL, MIT

Akamai Technologies

麻省理工学院数学系、计算机科学&人工智能实验室

阿卡迈科技

Albert R Meyer

Massachusets Institute of Technology

麻省理工学院

Copyright © 2010, Eric Lehman, F Tom Leighton, Albert R Meyer 。版权所有。

Contents 内容

I Proofs 证明

Introduction 引言

This text explains how to use mathematical models and methods to analyze problems that arise in computer science. The notion of a proof plays a central role in this work.

这本教材解释了怎样使用数学模型和方法分析计算机科学中的问题。证明的概念是本书的核心。

Simply put, a proof is a method of establishing truth. Like beauty, "truth" sometimes depends on the eye of the beholder, and it should not be surprising that what constitutes a proof differs among fields. For example, in the judicial system, legal truth is decided by a jury based on the allowable evidence presented at trial. In the business world, authoritative truth is specified by a trusted person or organization, or maybe just your boss.In fields such as physics and biology, scientific truth[1] is confirmed by experiment. In statistics, probable truth is established by statistical analysis of sample data.

简单来说；证明是证实**真理**的方法。像美一样，要说“真理”是什么，有时也见仁见智。但不足为奇的是，不同领域对证明的定义千差万别。例如，在司法系统中，**法律真实**是由陪审团基于可在审判时呈现的证据决定的。在企业界，**权威真实**是由可信的人或组织（也可能只是你的老板）指定的。在物理、生物之类的领域，**科学事实**[1]是由实验证实的。在统计学中，**或然性真理**是由采样数据的统计分析证实的。

Philosophical proof involves careful exposition and persuasion typically based on a series of small, plausible arguments. The best example begins with "Cogito ergo sum," a Latin sentence that translates as "I think, therefore I am." It comes from the beginning of a 17th century essay by the mathematician/philosopher, Rene´ Descartes, and it is one of the most famous quotes in the world: do a web search on the phrase and you will be flooded with hits.

哲学证明需要仔细的阐述和说服，而这些通常基于一系列微小又貌似合理的论据。有个绝佳的例子，以拉丁句子"Cogito ergo sum,"(翻译过来是“我思故我在”)打头。这句话出自 17 世纪数学家、哲学家勒奈·笛卡尔的一篇论文的开头，也是世界上最著名的引语：在网上搜索下该习语，检索结果会多到令你目不暇接。

Deducing your existence from the fact that you're thinking about your existence is a pretty cool and persuasive-sounding idea. However, with just a few more lines of argument in this vein, Descartes goes on to conclude that here is an infinitely beneficent God. Whether or not you believe in a beneficent God, you'll probably agree that any very short proof of God's existence is bound to be far-fetched. So even in masterful hands, this approach is not reliable.

从“你正在思考你的存在”这一事实来推断你的存在，这是个绝妙又令人信服的想法。然而，又用这种风格写了几行论据之后，笛卡尔接着得出结论，人世间存在一个无限仁慈的上帝。不管你是否信仰仁慈的上帝，你可能会同意，对上帝存在性的任何简短证明，肯定都是穿凿附会的。所以即使为大师所用，这一方法也并不可靠。

---

[1]Actually, only scientific falsehood can be demonstrated by an experiment——when the experiment fails to behave as predicted. But no amount of experiment can confirm that the next experiment won't fail. For this reason, scientists rarely speak of truth, but rather of theories that accurately predict past, and anticipated future, experiments.

[1]实际上，只有科学上的谬误才能用实验来证明——在实验无法像预测的那样进行时。但即使再多的实验也无法确保下一次实验不会失败。因此，科学家几乎从不说真理，而宁可说准确**推测过去**或**预测未来**的理论及实验。

Mathematics has its own specific notion of "proof".

数学有其特定的“证明”概念。

Definition. A mathematical proof of a *proposition* is a chain of *logical deductions* leading to the proposition from a base set of *axioms*.

定义： 命题的数学证明是，从基本**公理**集中导出该命题的**逻辑推理**链。

The three key ideas in this definition are highlighted: proposition, logical deduction, and axiom. These three ideas are explained in the following chapters, beginning with propositions in Chapter 1. We will then provide lots of examples of proofs and even some examples of "false proofs" (that is, arguments that look like a proof but that contain missteps, or deductions that aren't so logical when examied closely).False proofs are often even more important as examples than correct proofs, because they are uniquely helpful with honing your skills at making sure each step of a proof follows logically from prior steps.

我突出显示了该定义中的三个关键概念：**命题**、**逻辑推理**、**公理**。在下面的章节中会解释这三个概念，首先从第一章中的**命题**开始。我们会在那块内容中给出许多证明示例，甚至一些“错误证明“示例（也就是说，看起来像是证明，但却包含错误的**论据**；或仔细检查时，会发现不太合乎逻辑的**推理**）用作示例时，错误的证明常常比正确的证明更有价值。因为它们会神奇地帮你磨炼这方面的技巧——确保能从先前的步骤合乎逻辑地推导出证明的每一步。

Creating a good proof is a lot like creating a beautiful work of art. In fact, mathematicians often refer to really good proofs as being "elegant" or "beautiful".

构造一个好的证明很像是创造一件优美的艺术品。事实上，数学家常把绝佳的证明称作”优雅的“或”巧妙的“。

As with any endeavor, it will probably take a little practice before your fellow students use such praise when referring to your proofs, but to get you started in the right direction, we will provide templates for the most useful proof techniques in Chapter 2 and 3. We then apply these techniques in Chapter 4 to establish some important facts about numbers; facts that form the underpinning of one of the world's most widely-used cryptosystems.

像任何尝试一样，在你的同学这样赞美你的证明之前，你可能需要一些练习。但为了让你起步时就朝着正确的方向，我们会在第 2 章和第 3 章给出模板，用于传授最有用的证明技巧。然后我们将在第 4 章应用这些技巧来证实有关**数字**的一些重要事实；这些事实构成了一种世界上使用范围最广的密码系统的基础。

1 Propositions 命题

Definition. A proposition is a statement that is either true or false.

定义： 命题是只能为**真**或**假**的陈述。

For example, both of the following statements are propositions. The first is true and the second is false.

例如，以下两个陈述都是命题。第一个是真命题，而第二个是假命题。

Proposition 1.0.1. $2 + 3 = 5$.

Proposition 1.0.2. $1 + 1 = 3$.

命题 1.0.1. $2 + 3 = 5$.

命题 1.0.2. $1 + 1 = 3$.

Being true or false doesn't sound like much of a limitation, but it does exclude statements such as, "Wherefore art thou Romeo?" and "Give me an A!".

只能为真或假听起来算不上是限制，但它确实排除了类似的陈述——”为什么你偏偏是罗密欧呢？“及”给我打 A!“

Unfortunately, it is not always easy to decide if a proposition is true or false, or even what the proposition means. In part, this is because the English language is riddled with ambiguities. For example, consider the following statements:

1."You may have cake, or you may have ice cream."

2."If pigs can fly, then you can understand the Chebyshev bound."

3."If you can solve any problem we come up with, then you get an A for the course."

4."Every American has a dream."

不幸的是，确定命题的真假，乃至命题的含义，并非总是易事。这在某种程度上是因为，英语这种语言充满了歧义。例如，考虑以下陈述：

1.“你可以吃蛋糕或冰激凌。“

2.”如果猪会飞，那么你就能理解契比雪夫不等式。”

3.“如果你能解决我们提出的任何问题，那么这门课你就可以得 A。”

4.“每个美国人都有梦想。”

What precisely do these sentences mean? Can you have both cake and ice cream or must you choose just one dessert? If the second sentence is true, then is the Chebyshev bound incomprehensible? If you can solve some problems we come up with but not all, then do you get an A for the course? And can you still get an A even if you can't solve any of the problems? Does the last sentence imply that all Americans have the same dream or might some of them have different dreams?

这些句子的精确含义是什么？蛋糕和冰激凌这两种甜点你都能吃，还是必须二者择一？如果第二个句子为真，那么契比雪夫不等式就无人能懂吗？如果你只能解决我们提出的部分问题，那么这门课程你能得 A 吗？即使你不能解决任何问题，你是否依然能得 A? 最后一个句子暗示着所有美国人都有同一个梦想，还是有些美国人有不同的梦想?

Some uncertainty is tolerable in normal conversation. But when we need to formulate ideas precisely——as in mathematics and programming——the ambiguities inherent in everyday language can be a real problem. We can't hope to make an exact argument if we're not sure exactly what the statements mean. So before we start into mathematics, we need to investigate the problem of how to talk about mathematics.

日常会话中容许存在一些不确定性。但当我们想要精确地陈述观点时——就像在数学与编程中那样——日常语言固有的歧义就会真正成为问题。如果我们无法确认陈述的准确含义，就不要指望拿出确切的论据。所以在学习数学之前，需要搞清楚应该怎样谈论数学。

To get around the ambiguity of English, mathematicians have devised a special mini-language for talking about logical relationships. This language mostly uses ordinary English words and phrases such as "or", "implies", and "for all".But mathematicians endow these words with definitions more precise than those found in an ordinary dictionary. Without knowing these definitions, you might sometimes get the gist of statements in this language, but you would regularly get misled about what they really meant.

为规避英语的歧义，数学家设计了一种特殊的迷你语言来谈论逻辑关系。该语言大多采用普通的英语单词和词组，如：“或”('or')、“蕴涵”('implies')、“对于所有的“（'for all')。但数学家赋予这些单词的定义，比普通字典里的更精确。要是不了解这些定义，你可能偶尔能够领会用该语言所表达的陈述的要点，但肯定会经常误解它们的真实含义。

Suprisingly, in the midst of learning the language of mathematics, we'll come across the most important open problem in computer science——a problem whose solution could change the world.

令人惊奇的是，在学习数学语言时，我们将碰到计算机科学中最有价值的未决问题——该问题的解决可能会改变世界。

1.1 Compound Propositions 复合命题

In English, we can modify, combine, and relate propositions with words such as "not", "and", "or", "implies", and "if-then". For example, we can combine three propositions into one like this:

If all humans are mortal **and** all Greeks are human, **then** all Greeks are mortal.

在英语中，我们可以用如下的词修改、组合、关联命题——“非”、“且”、“或”、“蕴涵”、“如果-那么”。例如，可以把三个命题组合成类似下面的这个：

如果所有人都终归一死**且**希腊人是人，**那么**希腊人终究会死。

For the next while, we won't be much concerned with the internals of propositions—— whether they involve mathematics or Greek mortality —— but rather with how propositions are combined and related. So we'll frequently use variables such as P and Q in place of specific propositions such as "All humans are mortal" and "2 + 3 = 5". The understanding is that these variables, like propositions, can take on only the values T(true) and F(false). Such true/false variables are sometimes called Boolean variables after their inventor, George——you guessed it ——Boole.

接下来的这段时间，我们不会太过关注命题的内在含义——无论它们涉及数学还是希腊人的必死性——相反，我们会更关注怎样组合关联命题。所以我们经常使用像 P 和 Q 一样的变量来代替像“人终有一死”和"2 + 3 = 5"一样的具体命题。我们这样做的原因是，这些变量像命题一样，可取的值只有两个：T(真)和 F(假）。这种真/假变量有时叫做布尔变量——以其发明者的名字乔治·布尔（你猜对了）命名。

1.1.1 NOT, AND, and OR 非、与、或

We can precisely define these special words using truth tables. For example, if P denotes an arbitrary proposition, then the truth of the proposition "NOT(P)" is defined by the following truth table:

P	NOT(P)
T	F
F	T

可使用**真值表**精确定义这些关键词。例如，如果 P 代表某个任意命题，则命题“非 P"的取值由以下真值表定义：

P	非 P
T	F
F	T

The first row of the table indicates that when proposition P is true, the proposition "NOT(P)" is false. The second line indicates that when P is false, "NOT(P)" is true. This is probably what you would expect.

表格的第一行表明，命题 P 为真时，命题“非 P”为假。第二行表明，P 为假时，“非 P”为真。这可能是你所期望的。

In general, a truth table indicates the true/false value of a proposition for each possible setting of the variables.

真值表通常反映了在每个可能的变量取值下，命题的真/假值。

For example, the truth table for the proposition "P AND Q" has four lines, since the two variables can be set in four different ways:

P	Q	P AND Q
T	T	T
T	F	F
F	T	F
F	F	F

例如，命题“P 与 Q”的真值表有四行，因为可以用四种方式来设定两个变量的值：

P	Q	P 与 Q
T	T	T
T	F	F
F	T	F
F	F	F

According to this table, the proposition "P AND Q" is true only when P and Q are both true. This is probably the way you think about the word "and".

根据这个表格，只有 P 和 Q 都为真时，命题“P与Q”才为真。也许你也是这样考虑单词“**与**”的。

There is a subtlety in the truth table for "P OR Q":

P	Q	P OR Q
T	T	T
T	F	T
F	T	T
F	F	F

"P或Q"的真值表中有个细微差别：

P	Q	P 或 Q
T	T	T
T	F	T
F	T	T
F	F	F

The second row of this table says that "P OR Q" is true even if both P and Q are true. This isn't always the intended meaning of "or" in everyday speech, but this is the standard definition in mathematical writing. So if a mathematician says, "You may have cake, or you may have ice cream," he means that you could have both.

该表格的第二行表明，即使 P 和 Q 都为真，“P或Q"也为真。这并非总是日常会话中”或“的预期含义，但却是数学著作中的标准定义。所以如果某个数学家说，”你可以吃蛋糕，**或**冰淇淋。“ 他的意思是两种都可以吃。

If you want to exclude the possibility of both having and eating, you should use "exclusive-or"(XOR):

P	**Q**	**P XOR Q**
T	T	F
T	F	T
F	T	T
F	F	F

如果你想要排除把两种都拿来吃的可能性，应使用”异或“（XOR):

P	**Q**	**P 异或 Q**
T	T	F
T	F	T
F	T	T
F	F	F

1.1.2 IMPLIES 蕴涵

The least intuitive connecting word is "implies". Here is its truth table, with the lines labeled so we can refer to them later.

P	Q	P IMPLIES Q
T	T	T (tt)
T	F	F (tf)
F	T	T (ft)
F	F	T (ff)

最不直观的连接词是“蕴涵”。下面是它的真值表，每行都加了标记，以便在之后提及。

P	Q	P 蕴涵 Q
T	T	T (tt)
T	F	F (tf)
F	T	T (ft)
F	F	T (ff)

Let's experiment with this definition. For example, is the following proposition true or false?

"If the Riemann Hypothesis is true, then $x^2 \geq 0$ for every real number x."

让我们试用下该定义。例如，以下命题是真还是假?

"如果黎曼假设为真，那么对于每个实数 x,有$x^2 \geq 0$"。

The Riemann Hypothesis is a famous unresolved conjecture in mathematics —no one knows if it is true or false. But that doesn't prevent you from answering the question! This proposition has the form P IMPLIES Q where the *hypothesis*, P , is “the Riemann Hypothesis is true” and the *conclusion*, Q, is “$x^2 \geq 0$ for every real number x”. Since the conclusion is definitely true, we're on either line (tt) or line (ft) of the truth table. Either way, the proposition as a while is *true*!

黎曼假设是数学中一个尚未证实的著名猜想——没人知道它的真假。但这并不妨碍你回答该问题！这个命题有着“P 蕴涵 Q"的形式——假设 P 是”黎曼假设是真的“，而结论 Q 是”对于每个实数 x, $x^2 \geq 0$ “。因为结论肯定为真，所以该命题要么落在真值表的 tt 行，要么落在真值表的 ft 行。不管是哪种情况，该命题总是为真！

One of our original examples demonstrates an even stranger side of implications.

"If pigs can fly, then you can understand the Chebyshev bound."

我们最初的一个示例明显显露了**蕴涵**更为奇怪的一面。

”如果猪会飞，那么你就能理解契比雪夫不等式。“

Don't take this as an insult; we just need to figure out whether this proposition is true or false. Curiously, the answer has nothing to do with whether or not you can understand the Chebyshev bound. Pigs cannot fly, so we're on either line(ft) or line(ff) of the truth table. In both cases, the proposition is true!

别把它当侮辱；我们只需要搞清楚这一命题是真还是假。奇怪的是，答案和你是否能够理解契比雪夫不等式没有一点关系。猪不会飞，所以该命题落在真值表的 ft 行或 ff 行。在两种情形下，该命题都为真！

In contrast, here's an example of a false implication:

"If the moon shines white, then the moon is made of white cheddar."

相反，下面的示例是一个值为假的蕴涵：

"如果月亮发白光，那么它是用白色切达干酪做成的。"

Yes, the moon shines white. But, no, the moon is not made of white cheddar cheese. So we're on line(tf) of the truth table, and the proposition is false.

是的，月亮发白光。但是，月亮却不是用白色切达干酪做成的。所以该命题落在真值表的 tf 行，它的值为假。

The truth table for implications can be summarized in words as follows:

An implication is true exactly when the if-part is false or the then-part is true.

可以用如下的话来总结蕴涵的真值表：

当某蕴涵"**如果部分**"的值为**假**或"**那么部分**"的值为**真**时，该蕴涵的值正好为**真**。

This sentence is worth remembering; a large fraction of all mathematical statements are of the if-then form!

这句话值得牢记; 一大部分数学公式有着**如果-那么**的格式！

1.1.3 IFF 当且仅当

Mathematicians commonly join propositions in one additional way that doesn't arise in ordinary speech. The proposition "P if and only if Q" asserts that P and Q are logically equivalent; that is, either both are true or both are false.

数学家常用普通谈话中所没有的罕见方式来连接命题。命题"**P 当且仅当 Q**"断言 P 和 Q 逻辑上等价；即，**要么两者全为真；要么两者全为假**。

P	**Q**	**P IFF Q**
T	T	T
T	F	F
F	T	F
F	F	T

For example, the following if-and-only-if statement is true for every real number x:

例如，对于每个实数 x, 以下的**"当且仅当"陈述**都为真：

$x^2 - 4 \geq 0$ iff $|x| \geq 2$

For some values of x, both inequalities are true. For other values of x, neither inequality is true. In every case, however, the proposition as a whole is true.

对于 x 的某些值，两个不等式都为真。对于 x 的其他值，两个等式都不为真。但是不管怎样，**整个命题**都为真。

1.1.4 Notation 记法

Mathematicians have devised symbols to represent words like "AND" and "NOT". The most commonly-used symbols are summarized in the table below.

English	Symbolic Notation
NOT(P)	$\neg P$ (alternatively, $\overline{P}$)
P AND Q	P $\wedge$ Q
P OR Q	P $\vee$ Q
P IMPLIES Q	P $\rightarrow$ Q
if P then Q	P $\rightarrow$ Q
P IFF Q	P $\leftrightarrow$ Q

数学家设计了符号来代表“与”和“非”之类的词。下表总结了最常用的符号。

English	符号记法
非 P	$\neg P$ (或,$\overline{P}$)
P 与 Q	P $\wedge$ Q
P 或 Q	P $\vee$ Q
P 蕴涵 Q	P $\rightarrow$ Q
如果 P，那么 Q	P $\rightarrow$ Q
P 当且仅当 Q	P $\leftrightarrow$ Q

For example, using this notation, "If P AND NOT(Q), then R" would be written:

例如，使用这种记法，“如果 P 与(非 Q), 那么 R”将记作：

$$(P \wedge \overline{Q}) \rightarrow R$$

This symbolic language is helpful for writing complicated logical expressions compactly. But words such as "OR" and "IMPLIES" generally serve just as well as the symbols $\vee$ and $\rightarrow$, and their meaning is easy to remember. We will use the prior notation for the most part in this text, but you can feel free to use whichever convention is easiest for you.

这种符号语言能够简洁地标记复杂的逻辑表达式。不过，像“或”和“蕴涵”之类的词，作用通常和符号$\vee$ 和 $\rightarrow$一样，符号的含义还容易记。本书中的大部分内容都将使用前面的记法，但你可以不受限制地使用对你来讲最简单的任何记法。

1.1.5 Logically Equivalent Implications 逻辑等价蕴涵

Do these two sentences say the same thing?

If I am hungry, then I am grumpy.
If I am not grumpy, then I am not hungry.

这两句话说的是同一件事吗?

"**如果**我饿了，**那么**我会颓丧易怒。“

"**如果**我愉快平和，**那么**我不饿。“

We can settle the issue by recasting both sentences in terms of propositional logic. Let P be the proposition “I am hungry”, and let Q be “I am grumpy”. The first sentence says “P IMPLIES Q” and the second says “NOT(Q) IMPLIES NOT(P) ”.

用命题逻辑的术语重新组织这两个句子，可以解决该问题。设 P 为命题“我饿了”，Q 为命题“我颓丧易怒”。第一个句子表示为“P 蕴涵 Q”，第二个句子表示为“非(Q)蕴涵非(P)”。

We can compare these two statements in a truth table:

P	Q	P IMPLIES Q	NOT(Q) IMPLIES NOT(P)
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

我们可以在真值表中比较这两种陈述：

P	Q	P 蕴涵 Q	非Q 蕴涵 非P
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Sure enough, the columns of truth values under these two statements are the same, which precisely means they are equivalent. In general, "NOT(Q) IMPLIES NOT(P)" is called the **contrapositive** of the implication "P IMPLIES Q". And, as we've just shown, the two are just different ways of saying the same thing.

果不其然，这两个陈述的**真假值**列相同，这正好表明它们等价。“非 Q 蕴涵非 P"通常称为”P 蕴涵 Q"的对换命题。正如我们刚刚展示的那样，这两种陈述只是同一件事的不同描述方式。

In contrast, the converse of "P IMPLIES Q" is the statement "Q IMPLIES P". In terms of our example, the converse is:

If I am grumpy, then I am hungry.

相反，“P 蕴涵 Q”的逆命题是陈述“Q 蕴涵 P”。就我们的示例而言，逆命题是：

"**如果**我颓丧易怒，**那么**我会感觉饿。“

This sounds like a rather different contention, and a truth table confirms this suspicion:

P	Q	P IMPLIES Q	Q IMPLIES P
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

这听起来像是个相当不同的观点，而真值表证实了该推想：

P	Q	P 蕴涵 Q	Q 蕴涵 P
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

Thus, an implication is logically equivalent to its contrapositive but is not equivalent to its converse.

因此，蕴涵在逻辑上等价于它的对换命题，但却不等价于其逆命题。

One final relationship: an implication and its converse together are equivalent to an **iff statement**. For example,

If I am grumpy, then I am hungry, AND

if I am hungry, then I am grumpy.

are equivalent to the single statement:

I am grumpy IFF I am hungry.

最后一个关系：**蕴涵**"与上"其**逆命题**等价于**"当且仅当"陈述**。例如，

"**如果**我颓丧易怒，**那么**我会感觉饿。" **与上**

"**如果**我饿了，**那么**我会颓丧易怒。"

等价于单个陈述：

"我颓丧易怒**当且仅当**我饿了。"

Once again, we can verify this with a truth table:

P	Q	P IMPLIES Q	Q IMPLIES P	(P IMPLIES Q) AND (Q IMPLIES P)	P IFF Q
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

我们可以再次使用真值表来验证它：

P	Q	P 蕴涵 Q	Q 蕴涵 P	(P 蕴涵 Q) 与 (Q 蕴涵 P)	P 当且仅当 Q
T	T	T	T	T	T
T	F	F	T	F	F
F	T	T	F	F	F
F	F	T	T	T	T

1.2 Propositional Logic in Computer Programs 计算机程序中的命题逻辑

Propositions and logical connectives arise all the time in computer programs.For example, consider the following snippet, which could be either C, C++, or Java:

```
if( x>0 || (x<=0 && y>100) )
(further instructions)
```

计算机程序中始终存在**命题**和**逻辑连接词**。例如，考虑以下的代码片断，它可能是 C, C++或 Java 代码：

```
if( x>0 || (x<=0 && y>100) )
(附加指令)
```

The symbol || denotes "OR", and the symbol && denotes "AND". The further instructions are carried out only if the proposition following the word *if* is true. On closer inspection, this big expression is built from two simpler propositions. Let A be the proposition that x>0, and let B be the proposition that y>100. Then we can rewrite the condition "A OR (NOT(A) AND B)".

符号"||"代表”或“，而符号"&&"代表”与“。只有在单词 if 后面的命题为真时，才会执行附加指令。经过一番更为仔细的检查可知，这个大表达式由两个更简单的命题构成。设 A 为命题”**x>0**“，B 为命题”**y>100**“。那么我们就可以把该条件重写为”**A 或((非 A)与 B)**“

A truth table reveals that this complicated expression is logically equivalent to "A OR B".

A	**B**	**A OR (NOT(A) AND B)**	**A OR B**
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

真值表揭示了这一复杂的表达式，在逻辑上等价于”A 或 B“。

A	**B**	**A 或((非 A)与 B)**	**A 或 B**
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

This means that we can simplify the code snippet without changing the program's behavior:

if(x>0 || y>100)

(further instructions)

这意味着我们可以在不改变程序行为的情况下，简化代码片断：

if(x>0 || y>100)

(附加指令)

Rewriting a logical expression involving many variables in the simplest form is both difficult and important. Simplifying expressions in software can increase the speed of your program. Chip designers face a similar challenge——instead of minimizing && and || symbols in a program, their job is to minimize the number of analogous physical devies on a chip. The payoffs is potentially enormous: a chip with fewer devices is smaller, consumes less power, has a lower defect rate, and is cheaper to manufacture.

以最简单的形式重写涉及多个变量的逻辑表达式，虽然困难，却很有价值。简化软件里的表达式能够提升程序运行速度。芯片设计师面临着相似的挑战，他们的工作是最小化芯片上的模拟物理设备数目，而不是最小化程序中的“&&”和“||”符号数目。潜在的回报是巨大的：芯片上的设备数目越少，其体积就越小，耗能越少，次品率越低，造价越低。

1.3 Predicates and Quantifiers 谓词和量词

1.3.1 Propositions with Infinitely Many Cases 有无限多种情况的命题

Most of the examples of propositions that we have considered thus far have been straightforward in the sense that it has been relatively easy to determine if they are true or false. At worse, there were only a few cases to check in a truth table. Unfortunately, not all propositions are so easy to check. That is because some proposition may involve a large or infinite number of possible cases. For example, consider the following proposition involving prime numbers. (A prime is an integer greater than 1 that is divisible only by itself and 1. For example, 2,3,5,7, and 11

are primes, but 4, 6, and 9 are not. A number greater than 1 that is not prime is said to be *composite*.)

到目前为止，我们考虑过的大部分命题示例都是易懂的，意即要确定它们的真假是相对容易的。即便是在较糟糕的情形下，也只需在真值表中检查几种情况。不幸的是，不是所有命题都这样易于检查。因为有些命题可能包含大量或无限多种可能情况。例如，考虑以下涉及素数的命题。(素数是只能被自身和 1 整除的大于 1 的整数。例如，2,3,5,7, 11 是素数，但 4, 6, 9 不是。我们称大于 1 的非素数是**可分解的**。）

Proposition 1.3.1. For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.

命题 1.3.1. 对于每个非负整数 n，$n^2 + n + 41$ 的值是素数。

It is not immediately clear whether this proposition is true or false. In such circumstances, it is tempting to try to determine its veracity by computing the value of[1]

$$p(n) ::= n^2 + n + 41 \quad (1.1)$$

for several values of n and then checking to see if they are prime. If any of the computed values is not prime, then we will know that the proposition is false. If all the computed values are indeed prime, then we might be tempted to conclude that the proposition is true.

该命题的真假并非显而易见。在这样的情形下，用如下方式确定该命题真实与否就很吸引人了——对于 n 的一些值，计算出

$$p(n) ::= n^2 + n + 41 \quad (1.1)$$

的值[1]，然后检查它们是否是素数。若任何一个计算结果不是素数，则该命题为假。如果所有计算结果的确都是素数，我们可能就会坚定地得出'"该命题为真"的结论。

We begin the checking by evaluating $P(0) = 41$, which is prime. $P(1) = 43$ is also prime. So is $P(2) = 47$, $P(3) = 53$,...,and $P(20) = 461$, all of which are prime. Hmmm... It is starting to look like P(n) is a prime for every nonnegative integer n. In fact, continued checking reveals that

p(n) is prime for all $n \leq 39$. The proposition certainly does seem to be true.

首先计算$P(0) = 41$，它是素数。$P(1) = 43$ 也是素数。$P(2) = 47$, $P(3) = 53$,..., $P(20) = 461$也一样,所有这些都是素数。嗯...看起来对每个非负整数 n，P(n)都是素数。事实上，继续检查可知，对于所有$n \leq 39$，p(n)都是素数。该命题看来很可能为真。

But $p(40) = 40^2 + 40 + 41 = 41 \cdot 41$, which is not prime. So it's not true that the expression is prime for all nonnegative integers, and thus the proposition is false!

但是$p(40) = 40^2 + 40 + 41 = 41 \cdot 41$，这不是素数。所以对于所有非负整数，该表达式都是素数的论点并不正确，因此此命题为假。

Although surprising, this example is not as contrived or rare as you might suspect. As we will soon see, there are many examples of propositions that seem to be true when you check a few cases (or even many), but which turn out to be false. The key to remember is that you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set.

虽然令人惊奇，但和你想的不同，这并不是个造作或罕见的例子。我们将很快了解到，有好多示例, 在只检查几种（甚至许多）情况时似乎为真，但却被证明为假。关键是要记住，对于无限集合，你无法通过检查由其元素构成的有限集合（无论这个有限集合有多大），来核实针对该无限集合的断言。

Propositions that involve all numbers are so common that there is a special notation for them. For example, Proposition 1.3.1 can also be written as

$\forall n \in \mathbb{N}$. p(n) is prime. (1.2)

包含所有数字的命题太常见了，所以有个代表它们的专门符号。例如，命题 1.3.1 也可以记作：

$\forall n \in \mathbb{N}$. p(n) 是素数。 (1.2)

Here the symbol $\forall$ is read "for all". The symbol $\mathbb{N}$ stands for the set of nonnegative integers, namely, 0, 1, 2, 3, ...(ask your instructor for the complete list). The symbol "$\in$" is read as "is a member of," or "belongs to," or simply as "is in". The period after the $\mathbb{N}$ is just a separator between phrases.

这里的符号$\forall$读作“**对于所有的**”。符号$\mathbb{N}$代表**非负整数集**，即 0, 1, 2, 3, ...（向你的教员要完整列表）。符号"$\in$" 读作“是...的成员”，或“属于”，或仅仅是“在...里”。$\mathbb{N}$后面的句号只是词组之间的分隔符。

Here is another example of a proposition that, at first, seems to be true but which turns out to be false.

下面是另一个命题示例，乍看这个命题似乎为真，但却被证明是假命题。

---

[1]The symbol ::= means "equal by definition." It's always ok to simply write "=" instead of ::=, but reminding the reader that an equality holds by definition can be helpful.

[1]符号::=意为“按定义相等”。通常可以仅仅写作"="而不是"::="，但要提醒读者，按定义成立的等式可能会有所帮助。

Proposition 1.3.2. $a^4 + b^4 + c^4 = d^4$ has no solution when a, b, c, d are positive integers.

命题 1.3.2. 当 a, b, c, d 是正整数时，$a^4 + b^4 + c^4 = d^4$无解。

Euler (pronounced "oiler") conjectured this proposition to be true in 1769. It was checked by humans and then by computers for many values of a, b, c, and d over the next two centuries. Ultimately the proposition was proven false in 1987 by Noam Elkies. The solution he found was a = 95800;b = 217519;c = 414560; d = 422481. No wonder it took 218 years to show the proposition is false!

1769 年，欧拉(读作"oiler")猜想该命题为真。在接下来的两个世纪中，先是人工，后是计算机，用许多不同的 a,b,c,d 取值，检查了该命题。最终，Noam Elkies 于 1987 年证实了这是个假命题。他发现的解为：a = 95800;b = 217519;c = 414560; d = 422481。难怪数学家花了 218 年才证明这是个假命题！

In logical notation, Proposition 1.3.2 could the written,

命题 1.3.2 可以用逻辑标记法记作：

$$\forall a \in \mathbb{Z}^+ \forall b \in \mathbb{Z}^+ \forall c \in \mathbb{Z}^+ \forall d \in \mathbb{Z}^+ . a^4 + b^4 + c^4 \neq d^4.$$

Here,$\mathbb{Z}^+$ is a symbol for the positive integers. Strings of $\forall$'s are usually abbreviated for easier reading, as follows:

这里的$\mathbb{Z}^+$是个代表正整数的符号。为了便于阅读，这一排$\forall$通常简写如下：

$$\forall a, b, c, d \in \mathbb{Z}^+ . a^4 + b^4 + c^4 \neq d^4.$$

The following proposition is even nastier.

下面这个命题更难处理。

Proposition 1.3.3. $313(x^3 + y^3) = z^3$ has no solution when x,y,z $\in \mathbb{Z}^+$.

This proposition is also false, but the smallest counterexample values for x, y, and z have more than 1000 digits! Even the world's largest computers would not be able to get that far with brute force. Of course, you may be wondering why anyone would care whether or not there is a solution to $313(x^3 + y^3) = z^3$ where x, y, and z are positive integers. It turns out that finding solutions to such equations is important in the field of elliptic curves, which turns out to be important to the study of factoring large integers, which turns out (as we will see in Chapter 4) to be important in cracking commonly-used cryptosystems, which is why mathematicians went to the effort to find the solution with thousands of digits.

命题 1.3.3. 当 x,y,z $\in \mathbb{Z}^+$时，$313(x^3 + y^3) = z^3$没有解。

这也是个假命题，但 x,y,z 的最小反例值有 1000 多位数字！即便世界上最大型的计算机用蛮力也不可能达到那么多位。你自然会好奇为什么会有人关心，在 x,y,z 都为正整数时$313(x^3+y^3)=z^3$是否有解。原来找到**这类方程的解**，在**椭圆曲线**领域很有价值; 而人们发现椭圆曲线对研究分解大整数很重要；而**大整数的分解研究**又被证明在**破解常用密码系统**中很有价值（我们将在第四章学到该内容）。这就是为什么数学家耗费精力也要找到带有数千位数字的解。

Of course, not all propositions that have infinitely many cases to check turn out to be false. The following proposition (known as the "Four-Color Theorem") turns out to be true.

当然，不是所有需要检查无穷多种情况的命题都被证明是假命题。下面这个命题（称作“四色原理”）被证明是真命题。

Proposition 1.3.4. *Every map can be colored with 4 colors so that adjacent regions have different colors.*

The proof of this proposition is difficult and took over a century to perfect. Along the way, many incorrect proofs were proposed, including one that stood for 10 years in the late 19th century before the mistake was found. An extremely laborious proof was finally found in 1976 by mathematicians Appel and Haken, who used a complex computer program to categorize the four-colorable maps; the program left a few thousand maps uncategorized, and these were checked by hand by Haken and his assistants——including his 15-year-old daughter. There was a lot of debate about whether this was a legitimate proof: the proof was too big to be checked without a computer, and no one could guarantee that the computer calculated correctly, nor did anyone have the energy to recheck the four-colorings of the thousands of maps that were done by hand. Within the past decade, a mostly intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check the colorability of several hundred special maps.[3]

命题 1.3.4. *每幅地图都可用 4 种颜色着色，使得相邻区域颜色不同。*

该命题难以证明，历代数学家花费了一个多世纪来完善对它的证明。在此期间，数学家提出了许多错误的证明，其中有个 19 世纪末提出的证明，起初被认为是正确的，过了 10 年人们才发现错误。1976 年，数学家 Appel 和 Haken 终于完成了一个极其耗时费力的证明，他们使用复杂的计算机程序来归类四色地图；这个程序留下了几千张未分类的地图；Haken 与他的助手，还有他 15 岁的女儿，手工检查了这些地图。人们对这个证明是否合理有许多争论：该证明太过庞杂，不用计算机没法检查。没人能保证计算机不出差错，也没人有精力重新核查由手工检查过的几千幅地图的“四色”着色情况。在过去的 10 年中，有人提出了一个四色原理的证明，大部分内容明白易懂。可是仍然需要计算机来检查几百幅特殊地图的着色情况。[3]

In some cases, we do not know whether or not a proposition is true. For example, the following simple proposition (known as Goldbach's Conjucture) has been heavily studied since 1742 but we still do not know if it is true. Of course, it has been checked by computer for many values of n, but as we have seen, that is not sufficient to conclude that it is true.

在有些情况下，我们不知道某个命题是否为真。例如，从 1742 年起，数学家就开始费力研究下面这个简单命题（称作哥德巴赫猜想），但迄今为止，我们仍不知道它是否为真。当然，计算机已经核实过许多 n 值。但正如我们所知道的，这并不足以推断出该命题为真。

Proposition 1.3.5 (Goldbach). Every even integer n greater than 2 is the sum of two primes.

命题 1.3.5（哥德巴赫）每个大于 2 的偶数 n 都是两素数之和。

While the preceding propositions are important in mathematics, computer scientists are often interested in propositions concerning the "correctness" of programs and systems, to determine whether a program or system does what it's supposed to do. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip

correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

虽然前述命题在数学中很重要，但经常使计算机科学家发生兴趣的是，与程序和系统“正确性”相关的命题，该命题可用来确定某个程序或系统是否实现了设计功能。众所周知，程序中藏有许多缺陷。越来越多的研究者及从业者群体，正努力寻找证明程序正确性的方法。这些尝试在 CPU 芯片领域已颇具成效，所以现在主要的芯片制造商通常使用它们来证明芯片的正确性，避免出现与以下类似的错误——如 20 世纪 90 年代臭名昭著的**奔腾浮点除错误**。

Developing mathematical methods to verify programs and systems remains an active research area. We'll consider some of these methods later in the text.

发展数学方法来验证程序和系统仍是个活跃的研究领域。我们将在后面的章节中考虑其中一些方法。

---

[3] See http://www.math.gatech.edu/~thomas/FC/fourcolor.html

The story of the Four-Color Proof is told in a well-reviewed popular (non-technical) book: "Four Colors Suffice. How the Map Problem was Solved." *Robin Wilson*. Princeton Univ. Press,2003, 276pp. ISBN 0-691-11533-8.

[3]请参阅 http://www.math.gatech.edu/~thomas/FC/fourcolor.html

在颇受好评的通俗书籍（非技术的）《四种颜色就够了——怎样解决地图问题》里讲了证明四色原理的故事。*Robin Wilson*. Princeton Univ. Press,2003, 276pp. ISBN 0-691-11533-8.

1.3.2 Predicates 谓词

A predicate is a proposition whose truth depends on the value of one or more variables. Most of the propositions above were defined in terms of predicates. For example,

"n is a perfect square"

is a predicate whose truth depends on the value of n. The predicate is true for n = 4 since four is a perfect sqaure, but false for n=5 since five is not a perfect square.

真实性依赖于一个或多个变量的命题叫**谓词**。以上大多数命题的定义都与谓词相关。例如，

"n 是完全平方数"

是谓词，它的真实性依赖于 n 的值。因为 4 是完全平方数，所以 n=4 时，该谓词为真；而 5 不是完全平方数；所以 n=5 时该谓词为假。

Like other propositions, predicates are often named with a letter. Furthermore, a function-like notation is used to denote a predicate supplied with specific variable values. For example, we might name our earlier predicate P:

P(n)::="n is a perfect square"

Now P(4) is true, and P(5) is false.

像其他命题一样，谓词常用字母命名。此外，我们还使用类似函数的符号来标记需要指定变量值的谓词。例如，我们可能会把之前的谓词命名为 P:

P(n)::="n 是个完全平方数"

现在 P(4)为真, 而 P(5)为假.

This notation for predicates is confusingly similar to ordinary function notation. If P is a predicate, then P(n) is either true or false, depending on the value of n. On the other hand, if p is an ordinary function, like $n^2 + n$, then p(n) is a *numerical quantity*. **Don't confuse these two!**

令人困惑的是，代表谓词的符号与普通函数的符号相似。如果 P 是谓词，那么根据不同的 n 值，P(n)要么为真，要么为假。另一方面，如果 p 是像 $n^2 + n$一样的普通函数，那么 p(n)是*数值量*。**别把这两个弄混了！**

1.3.3 Quantifiers 量词

There are a couple of assertions commonly made about a predicate: that it is sometimes true and that it is always true. For example, the predicate

"$x^2 \geq 0$"

is always true when x is a real number. On the other hand, the predicate

"$5x^2 - 7 = 0$"

is only sometimes true; specifically, when $x = \pm\sqrt{\frac{7}{5}}$.

我们常对谓词下两种断言：**有时为真**与**始终为真**。例如，当 x 是实数时，谓词

"$x^2 \geq 0$"

始终为真。另一方面，谓词

"$5x^2 - 7 = 0$"

只是**有时为真**；具体来讲，是在 $x = \pm\sqrt{\frac{7}{5}}$时为真。

There are several ways to express the notions of "always true" and "sometimes true" in English. The table below gives some general formats on the left and specific examples using those formats on the right. You can expect to see such phrases hundreds of times in mathematical writing!

英语中有几种方式可表达“**始终为真**”及“**有时为真**”的概念。下表左侧列出了一些通用格式，右侧列出了使用这些格式的具体示例。你可能会在数学著作中无数次看到这类短语!

Always True

For all n,P(n) is true.	**For all** $x \in R, x^2 \geq 0.$
P(n) is true for every n.	$x^2 \geq 0$ **for every** $x \in R.$

始终为真

对于所有的 n,P(n)为真。	**对于所有的** $x \in R, x^2 \geq 0$**。**
对于每个 n,P(n)为真。	**对于每个** $x \in R$**，** $x^2 \geq 0$**。**

Sometimes True

There exists an n such that P(n) is true.	**There exists an** $x \in \mathbb{R}$ **such that** $5x^2 - 7 = 0$**.**
P(n) is true for some n.	$5x^2 - 7 = 0$ **for some** $x \in \mathbb{R}$**.**
P(n) is true for at least one n.	$5x^2 - 7 = 0$ **for at least one** $x \in \mathbb{R}$**.**

有时为真

存在某个 n,使得 P(n)为真。	**存在某个** $x \in \mathbb{R}$**，使得**$5x^2 - 7 = 0$**。**
对于某个 n,P(n)为真。	**对于某个** $x \in \mathbb{R}$**，** $5x^2 - 7 = 0$**。**
对于至少一个 n,P(n)为真。	**对于至少一个** $x \in \mathbb{R}$**，** $5x^2 - 7 = 0$**。**

All these sentences quantify how often the predicate is true. Specifically, an assertion that a predicate is always true, is called a *universally quantified statement*. An assertion that a predicate is sometimes true, is called an *existentially quantified statement*.

所有这些句子**用量词限定了**谓词的频率。具体来讲，谓词**始终为真**的断言称作*全称量化陈述*。谓词**有时为真**的断言称作*存在量化陈述*。

Sometimes English sentences are unclear about quantification:

"If you can solve any problem we come up with, then you get an A for the course."

The phrase "you can solve any problem we can come up with" could reasonably be interpreted as either a universal or existential statement. It might mean:

"You can solve every problem we come up with,"

or maybe

"You can solve at least one problem we come up with."

有时英语句子里的量化并不清晰：

“如果你能解决我们提出的**任何**问题，那么这门课程你就能得 A。”

我们可以把“你能解决我们提出的**任何**问题”这一措辞，合理地解释为**全称量化陈述**或**存在量化陈述**。它的含义可能是：

“你能解决我们提出的**每个**问题，”

或者也可能是：

“你**至少**能解决我们提出的**一个**问题。”

In the preceding example, the quantified phrase appears inside a larger if-then statement. This is quite normal; quantified statements are themselves propositions and can be combined with AND OR IMPLIES, etc., just like any other proposition.

在前面的例子里，量化短语出现在更大的“如果-那么”陈述中。这很正常；量化陈述本身就是命题，因此可以像对待任何其他命题那样，用“与”、“或”、“蕴涵”等来组合它们。

1.3.4 More Notation 更多符号

There are symbols to represent universal and existential quantification, just as there are symbols for "AND"($\wedge$), "IMPLIES"($\rightarrow$), and so forth. In particular, to say that a predicate, P(x), is true for all values of x in some set, D, we write:

$$\forall x \in D.\, P(x) \quad (1.3)$$

数学中有代表**全称量化**与**存在量化**的符号，就像有着代表“与”（$\wedge$),"蕴涵“（$\rightarrow$)等的符号一样。具体来讲，为了表示对于某集合 D 里所有的 x 值，谓词 P(x)都为真，我们这样写：

$\forall x \in D. P(x)$ (1.3)

The universal quantifier symbol ∀ is read "for all", so this whole expression (1.3) is read "For all x in D, P(x) is true." Remember that upside-down "A" stands for "**A**ll".

全称量化符号 ∀读作“对于所有的”，所以整个表达式（1.3）读作“对于 D 里的所有 x,P(x)为真。”要记住倒置的“A”代表“所有的”("All")。

To say that a predicate P(x) is true for at least one value of x in D, we write:

$\exists x \in D. P(x)$ (1.4)

为了表示对于 D 中的至少一个 x 值，谓词 P(x)为真，我们这样写：

$\exists x \in D. P(x)$ (1.4)

The existential quantifier symbol ∃, is read "there exists". So expression (1.4) is read, "There exists an x in D such that P(x) is true." Remember that backward "E" stands for "**E**xists".

存在量化符号 ∃读作“存在”。所以表达式（1.4）读作，“D 中存在 x,使得 P(x)为真。” 记住，反向的"E"代表”存在“（"Exists")。

The symbols ∀ and ∃ are always followed by a variable——typically with an indication of the set the variable ranges over——and then a predicate, as in the two examples above.

如同上述的两个例子一样，符号 ∀和 ∃ 后面总是跟着变量——通常还带有标识符， 代表变量变化范围的集合，——然后是谓词。

As an example, let Probs be the set of problems we come up with, Solves(x) be the predicate "You can solve problem x", and G be the proposition, "You get an A for the course". Then the two different interpretations of

"If you can solve any problem we come up with, then you get an A for the course."

can be written as follows:

$(\forall x \in Probs.Solves(x))$ IMPLIES *G*,

or maybe

$(\exists x \in Probs.Solves(x))$ IMPLIES *G*.

举个例子，设*Probs*为我们所提问题的集合，设*Solves(x)*为谓词"你能解决问题 x"，设*G*为命题，"你这门课得 A"。那么

"如果你能解决我们提出的**任何**问题，那么这门课程你就能得 A。"

的两种不同解释标记如下：

$(\forall x \in Probs.Solves(x))$ 蕴涵 *G*,

或者也可能是

$(\exists x \in Probs.Solves(x))$ 蕴涵 *G*.

1.3.5 Mixing Quantifiers 组合量词

Many mathematical statements involve several quantifiers. For example, *Goldbach's Conjecture* states:

"Every even integer greater than 2 is the sum of two primes."

Let's write this more verbosely to make the use of quantification clearer:

For every even integer n greater than 2, there exist primes p and q such that $n = p + q$.

Let Evens be the set of even intergers greater than 2, and let Primes be the set of primes. Then we can write Goldbach's Conjecture in logic notation as follows:

$$\underbrace{\forall n \in Evens.}_{\text{for every even integer n>2}} \quad \underbrace{\exists p \in Primes\ \exists q \in Primes.}_{\text{there exist primes p and q such that}} n = p + q$$

The proposition can also be written more simply as

$\forall n \in Evens. \exists p, q \in Primes. p + q = n.$

许多数学陈述包含几个量词。例如，哥德巴赫猜想称：

“每个大于 2 的偶数都是两素数之和。“

把这句话写得长一些，以使量化的用法更清晰：

对于每个大于 2 的偶数，存在素数 p、q, 使得$n = p + q.$

设*Evens*为大于 2 的偶数集合，设*Primes*为素数集合。则可用如下的逻辑符号来表示哥德巴赫猜想：

$$\underbrace{\forall n \in Evens.}_{\text{对于每个偶数n}>2}\underbrace{\exists p \in Primes\, \exists q \in Primes.}_{\text{存在素数p和q使得}} n = p + q$$

也可把该命题简化为

$\forall n \in Evens. \exists p, q \in Primes. p + q = n.$

1.3.6 Order of Quantifiers 量词的顺序

Swapping the order of different kinds of quantifiers(existential or univeral) usually changes the meaning of a proposition. For example, let's return to one of our initial, confusing statements:

"Every American has a dream."

This sentence is ambiguous because the order of quantifiers is unclear. Let A be the set of Americans, let D be the set of dreams, and define the predicate *H(a,d)* to be "American a has dream d." Now the sentence could mean that there is a single dream that every American shares:

$\exists d \in D. \forall a \in A. H(a, d)$

For example, it might be that every American shares the dream of owning their own home.

Or it could mean that every American has a personal dream:

$$\forall a \in A. \exists d \in D. H(a, d)$$

For example, some Americans may dream of a peaceful retirement, while others dream of continuing practicing their profession as long as they live, and still others may dream of being so rich they needn't think at all about work.

Swapping quantifiers in Goldbach's Conjecture creates a patently false statement; namely that every even number ≥ 2 is the sum of the same two primes:

$$\underbrace{\exists p, q \in Primes.}_{\substack{\text{there exist primes} \\ \text{p and q such that}}} \underbrace{\forall n \in Evens.}_{\substack{\text{for every even} \\ \text{integer n>2}}} n = p + q.$$

交换不同种类量词的顺序，通常会改变命题含义。例如，回到最初那个令人疑惑的陈述：

"每个美国人都有梦想。"

这句话有歧义，因为量词的顺序不清楚。设 A 为美国人的集合，D 为梦想的集合，定义谓词*H(a,d)*为"美国人 a 有梦想 d"。该句子的意思可能是，每个美国人都有同一个梦想。

$$\exists d \in D. \forall a \in A. H(a, d)$$

例如，可能是这种情形：拥有属于自己的住宅，是每个美国人都有的梦想。

或者也可以把它解释为，每个美国人都有个**个人梦想**：

$$\forall a \in A. \exists d \in D. H(a, d)$$

例如，有些美国人可能梦想着平静的退休生活；而其他人则梦想着，只要一息尚存，就会继续投身于事业；还有人可能梦想成为巨富，这样他们根本就不用考虑工作了。

交换哥德巴赫猜想的量词会形成明显错误的陈述；即每个偶数≥ 2都是同样的两个素数之和。

$$\underbrace{\exists p, q \in Primes.}_{\text{存在素数 p和q使得}}\underbrace{\forall n \in Evens.}_{\text{对于每个偶数 n>2}} n = p + q.$$

1.3.7 Variables Over One Domain 同一个域内的变量

When all the variables in a formula are understood to take values from the same nonempty set, D, it's conventional to omit mention of D. For example, instead of $\forall x \in D\ \exists y \in D.\, Q(x, y)$ we'd write $\forall x \exists y.\, Q(x, y)$. The unnamed nonempty set that x and y range over is called the domain of discourse, or just plain domain, of the formula.

当公式中的所有变量默认都从同一个非空集 D 里取值时，通常省略 D。例如，我们写$\forall x \exists y.\, Q(x, y).$，而非记作$\forall x \in D\ \exists y \in D.\, Q(x, y)$。限定 x、y 取值范围的那个未指定非空集，叫作公式的**论域**，或**普通域**。

It's easy to arrange for all the variables to range over one domain. For example, Goldbach's Conjecture could be expressed with all variables ranging over the domain $\mathbb{N}$ as

$\forall n.\, (n \in Evens)$ IMPLIES ($\exists p \exists q.\, p \in Primes$ AND $q \in Primes$ AND $n = p + q$).

把在同一个域内变化的所有变量排列起来并非难事。例如，哥德巴赫猜想的所有变量都处在域$\mathbb{N}$中，可将其表示为：

$\forall n.\, (n \in Evens)$ 蕴涵 ($\exists p \exists q.\, p \in Primes$ 与 $q \in Primes$ 与 $n = p + q$).

1.3.8 Negating Quantifiers 否定量词

There is a simple relationship between the two kinds of quantifiers. The following two sentences mean the same thing:

It is not the case that everyone likes to snowboard.

There exists someone who does not like to snowboard.

In terms of logic notation, this follows from a general property of predicate formulas:

NOT($\forall x.\ P(x)$) is equivalent to $\exists x$.NOT(P(x)).

那两种量词有着简单的关系。下面这两个句子说的是同一件事：

并非每个人都喜欢滑雪。

有人不喜欢滑雪。

——该论断是从谓词公式的一般特性得出的（用逻辑符号表示）：

非($\forall x.\ P(x)$) 等价于 $\exists x$.非(P(x)).

Similarly, these sentences mean the same thing:

There does not exist anyone who likes skiing over magma.

Everyone dislike skiiing over magma.

We can express the equivalence in logic notation this way:

NOT($\exists x.\ P(x)$) IFF $\forall x.$ NOT(P(x)) (1.5)

类似地，这两个句子也说的是同一件事：

没人喜欢在岩浆上滑雪。

所有人都不喜欢在岩浆上滑雪。

可用如下的逻辑符号表示该等价关系：

非($\exists x.\ P(x)$) 当且仅当 $\forall x.$ 非(P(x)) (1.5)

The general principle is that moving a "not" across a quantifier changes the kind of quantifier.

通用原则是，把”非“移到量词上会改变该量词的类型。

1.4 Validity 有效性

A propositional formula is called valid when it evaluates to T no matter what truth values are assigned to the individual propositional variables. For example, the propositional version of the Distributive Law is that P AND (Q OR R) is equivalent to (P AND Q) OR (P AND R). This is the same as saying that

[P AND (Q OR R)] IFF [(P AND Q) OR (P AND R)] (1.6)

is valid. This can be verified by checking the truth table for P AND (Q OR R) and (P AND Q) OR (p AND R):

P	Q	R	P AND (Q OR R)	(P AND Q) OR (P AND R)
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	F	F
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F

不管给**命题公式**里的单个命题变量赋什么真假值，最后求得的值总为T, 就称**该命题公式**有效。例如，用命题形式来描述分配律是, P 与 (Q 或 R) 等价于 (P 与 Q) 或 (P 与 R). 这等同于说

[P 与 (Q 或 R)] 当且仅当 [(P 与 Q) 或 (P 与 R)] （1.6）

是有效的。检查 P 与 (Q 或 R) 和 (P 与 Q) 或 (P 与 R)的真值表可证明这一点。

P	Q	R	P 与 (Q 或 R)	(P 与 Q) 或 (P 与 R)
T	T	T	T	T
T	T	F	T	T
T	F	T	T	T
T	F	F	F	F
F	T	T	F	F
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F

The same idea extends to predicate formulas, but to be valid, a formula now must evaluate to true no matter what values its variables may take over any unspecified domain, and no matter what interpretation a predicate variable may be given. For example, we already observed that the rule for negating a quantifier is captured by the valid assertion (1.5).

同样的思想可用在谓词公式上，但无论**公式的变量**取任何未指定域内的任何值，也无论怎样解释该**谓词变量**，求得的**公式值**必须为真，该**公式**才是有效的。例如，我们已经看到，有效断言(1.5)阐述了否定量词的规则。

Another useful example of a valid assertion is

$$\exists x \forall y.\, P(x,y) \text{ IMPLIES } \forall y \exists x.\, P(x,y). \qquad (1.7)$$

Here's an explanation why this is valid:

Let D be the domain for the variables and P_0 be some binary predicate[4] on D. We need to show that if

$$\exists x \in D \;\; \forall y \in D.\, P_0(x,y) \qquad (1.8)$$

holds under this interpretation, then so does

$$\forall y \in D \ \exists x \in D. P_0(x, y) \qquad (1.9)$$

So suppose (1.8) is true. Then by definition of ∃, this means that some element $d_0 \in D$ has the property that

$$\forall y \in D. P_0(d_0, y).$$

By definition of ∀, this means that

$$P_0(d_0, d)$$

is true for all $d \in D$. So given any $d \in D$, there is an element in D, namely, d_0, such that $P_0(d_0, d)$ is true. But that's exactly what (1.9) means, so we've proved that (1.9) holds under this interpretation, as required.

另一个有用的有效断言示例是

$$\exists x \forall y. P(x, y) \text{ 蕴涵 } \forall y \exists x. P(x, y). \qquad (1.7)$$

下面给出了**证明该断言有效**的解释：

设D为变量的域，P_0是D上的某个二元谓词[4]。我们需要证明如果

$$\exists x \in D \ \forall y \in D. P_0(x, y) \qquad (1.8)$$

在该解释下成立，那么

$$\forall y \in D \ \exists x \in D. P_0(x, y) \qquad (1.9)$$

也成立。

假定(1.8)为真。然后根据 ∃的定义，这意味着某个元素$d_0 \in D$ 有如下特性：

$$\forall y \in D. P_0(d_0, y).$$

根据 ∀的定义，这意味着对于所有的 $d \in D$，都有

$$P_0(d_0, d)$$

为真。所以给定任意$d \in D$，都有一个D中的元素，即d_0，使得$P_0(d_0, d)$为真。而这正是(1.9)的含义，所以我们已经按要求证明了(1.9)在该解释下成立。

We hope this is helpful as an explanation, although purists would not really want to call it a "proof". The problem is that with something as basic as (1.7), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (1.7) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification.

即使纯粹主义者实际上不想称其为“证明”，我们也希望这是个有用的解释，问题是，难以找到合适的更为基本的公理，来证明像（1.7）这样基础的东西。该解释所做的事情，就是把逻辑公式(1.7)翻译为英语，然后按照英语中“对于所有的”和“存在”的含义，来进行合理解释。

In contrast to (1.7), the formula

$$\forall y \exists x.\, P(x, y) \text{ IMPLIES } \exists x \forall y.\, P(x, y). \quad (1.10)$$

is not valid. We can prove this by describing an interpretation where the hypothesis, $\forall y \exists x.\, P(x, y)$, is true but the conclusion, $\exists x \forall y.\, P(x, y)$, is not true. For example, let the domain be the integers and P(x,y) mean $x > y$. Then the hypothesis would be true because, given a value, n, for y we could, for example, choose the value of x to be $n + 1$. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

与(1.7)相反，公式

$$\forall y \exists x.\, P(x, y) \text{ 蕴涵 } \exists x \forall y.\, P(x, y). \quad (1.10)$$

是无效的。通过描述一段解释可以证明这一点，在该解释中，假设$\forall y \exists x.\, P(x, y)$为真，但结论$\exists x \forall y.\, P(x, y)$为假。例如，设域为整数，而P(x,y)的含义是 $x > y$。那么该假设可能为真，因为，比方说，为y指定某个值n，我们可以选择把n+1作为x的值。但该解释的结论却断言，存在某个大

于所有整数的整数，这必定为假。类似这样证明断言为假的**解释**，称为该断言的**反模式**。

---

[4]That is, a predicate that depends on two variables.

[4] 也就是依赖两个变量的谓词。

1.5 Satisfiability 可满足性

A proposition is satisfiable if some setting of the variables makes the proposition true. For example, P AND $\overline{Q}$ is satisfiable because the expression is true if P is true **or** Q is false. On the other hand, P AND $\overline{P}$ is not satisfiable because the expression as a whole is false for both settings of P. But determining whether or not a more complicated proposition is satisfiable is not so easy. How about this one?

$$(P \text{ OR } Q \text{ OR } R) \text{ AND } (\overline{P} \text{ OR } \overline{Q}) \text{ AND } (\overline{P} \text{ OR } \overline{R}) \text{ AND } (\overline{R} \text{ OR } \overline{Q})$$

如果**某一组变量**使得命题为真，则认为该命题是**可满足的**。例如，“P 与 $\overline{Q}$”是可满足的，因为如果**P为真且Q为假**，则该表达式为真。（**译者注：**原文中误用了“或”）另一方面，“P 与 $\overline{P}$ ”是**不可满足的**。因为对于P的两个取值，整个表达式的值都为假。但要确定某个更为复杂的命题**是否可满足**并非易事。这个命题是否可满足?

$$(P \text{ 或 } Q \text{ 或 } R) \text{ 与 } (\overline{P} \text{ 或 } \overline{Q}) \text{ 与 } (\overline{P} \text{ 或 } \overline{R}) \text{ 与 } (\overline{R} \text{ 或 } \overline{Q})$$

The general problem of deciding whether a proposition is satisfiable is called *SAT*. One approach to SAT is to construct a truth table and check whether or not a **T** ever appears. But this approach is not very efficient; a proposition with n variables has a truth table with 2^n lines, so the effort required to decide about a proposition grows exponentially with the number of variables. For a proposition with just 30 variables, that's already over a billion lines to check!

判定**某命题是否可满足**的通用问题称为*SAT*。SAT的一种解决方法是创建真值表，并检查**T**到底有没有出现。但该方法并不太高效；含有n个变量的命题，它的真值表有 2^n 行。所以判定命题所花的工夫，会随变量数目的增多呈指数增长。对于一个只有30个变量的命题来讲，待检查的内容就已经超过10亿行了！

Is there a more efficient solution to SAT? In particular, is there some, presumably very ingenious, procedure that **determines** in a number of steps that grows polynomially——like n^2 or n^{14}——instead of exponentially, **whether any given proposition is satisfiable or not**? No one knows. And an awful lot hangs on the answer. An efficient solution to SAT would immediately imply efficient solutions to many, many other important problems involving packing, scheduling, routing, and circuit verification, among other things. This would be wonderful, but there would also be worldwide chaos. Decrypting coded messages would also become an easy task (for most codes). Online financial transactions would be insecure and secret communications could be read by everyone.

是否有更加高效的SAT解法？尤其是，有没有一个可能非常巧妙的步骤，用来确定任意给定命题是否可满足，并且所花的步骤数以多项式形式增长——如n^2 或 n^{14}，而非成指数级递增？没有人知道。这个答案至关重要。高效的SAT解法可能直接意味着无数其他重要问题都有了高效解决方案，这些问题包括打包、调度、路由、电路验证等等。这可能会很美好，但也可能导致世界范围的混乱。解密编码信息也会成为一项容易的任务（对大多数代码是这样的）。在线财务交易将会不安全，人人都可以读取私密通信。

Recently there has been exciting progress on *sat-solvers* for practical applications like digital circuit verification. These programs find satisfying assignments with amazing efficiency even for formulas with millions of variables. Unfortunately, it's hard to predict which kind of formulas are amenable to sat-solver methods, and for formulas that are NOT satisfiable, sat-solvers generally take exponential time to verify that.

用在数字电路验证之类实用应用上的sat解答器，最近取得了激动人心的进展。即使对于含有数百万个变量的公式，这些程序也能以惊人的效率，找到满足命题的一组变量赋值。不幸的是，难以预测哪些公式适合用sat解答器方法来求解，对于不可满足的公式，sat解答器通常要花指数级时间来验证。

So no one has a good idea how to solve SAT in polynomial time, or how to prove that it can't be done —— researchers are completely stuck. The problem of determining whether or not SAT has a polynomial time solution is known as the "**P** vs. **NP**" problem. It is the outstanding unanswered question in theoretical computer science. It is also one of the seven Millenium Problems: the Clay Institute will award you $1,000,000 if you solve the **P** vs. **NP** problem.

没人能想出在多项式时间内解答SAT（或证明做不到）的好主意——研究人员一筹莫展。判定SAT是否有多项式时间解的问题，称为“**P** vs. **NP**"问题。它是理论计算机科学中还未解决的重要问题。它也是七个千禧年难题之一：如果你解决了**P** vs. **NP**问题，克雷数学研究所将奖励你一百万美元。

2 Patterns of Proof 证明的方法

2.1 The Axiomatic Method 公理化方法

The standard procedure for establishing truth in mathematics was invented by Eu- clid, a mathematician working in Alexandria, Egypt around 300 BC. His idea was to begin with five assumptions about geometry, which seemed undeniable based on direct experience.For example, one of the assumptions was "There is a straight line segment between every pair of points." Prorallopositions like these that are simply accepted true are called axioms.

数学家欧几里得，生活于公元前 300 年左右的埃及亚历山大港，他发明了在数学中证实命题正确性的标准步骤。他的思想发端于几何学上的五个假设，它们基于直接经验，似乎毋庸置疑。例如，其中一个假设是“两点之间只有一个直线段。”像这样不证自明的命题称为公理。

Starting from these axioms, Euclid established the truth of many additional propositions by providing "proofs".A proof is a sequence of logical deductions **from** axioms and previously-proved statements that concludes with the proposition in question.

从这些公理开始，通过提供“证明”，欧几里得证实了许多附加命题的正确性。证明是一系列逻辑推论——从公理和之前证实过的陈述开始，以所讨论的命题作为结束。

There are several common terms for a proposition that has been proved.The different terms hint at the role of the proposition within a larger body of work.

- Important propositions are called *theorems*.
- A *lemma* is a preliminary proposition useful for proving later propositions.
- A *corollary* is a proposition that follows in just a few logical steps from a lemma or a theorem.

可用多个常见术语来指代已证明的命题。不同的术语暗示着该命题在更大工作体系中的作用。

- 重要的命题称作**定理**。
- **引理**是个可用于证明后续命题的初级命题。
- 从某个引理或定理开始，仅推导几步，就可得到叫做**推论**的命题。

Euclid's axiom-and-proof approach, now called the axiomatic method, is the foundation for mathematics today. In fact, just a handful of axioms, collectively called Zermelo-Frankel Set Theory with Choice(ZFC), together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics.

欧几里德的公理&证明方法，现在称为公理化方法，是现代数学的基石。事实上，只需几个公理——它们统称为包括选择公理的策梅洛-弗兰克尔集合论——加上一些逻辑推论规则，似乎就足以推导出大体上所有的数学理论。

2.1.1 Our Axioms 我们的公理

The ZFC axioms are important in studying and justifying the foundations of mathematics, but for practical purposes, they are much too primitive.Proving theorems in ZFC is a little like writing programs in byte code instead of a full-fledged programming language——by one reckoning, a formal proof in ZFC that 2 + 2 = 4 requires more than 20,000 steps! So instead of starting with ZFC, we're going to take a huge set of axioms as our foundation: we'll accept all familiar facts from high school math!

在研究及证明数学基础的合理性方面，ZFC 公理价值很大，但对实际应用来讲，它们却太过简陋。用 ZFC 证明定理有点象用字节码而非完备的编程语言来编写程序——据估计，用规范的 ZFC 来证明 2 + 2 = 4 需要超过 20,000 个步骤！所以与其从 ZFC 开始，倒不如把超大的公理集当作我们的基础：我们将把高中数学中的常见事实都默认为公理！

This will give us a quick launch, but you may find this imprecise specification of the axioms troubling at times.For example, in the midst of a proof, you may find yourself wondering, "Must I prove this little fact or can I take it as an axiom?"Feel free to ask for guidance, but really there is no absolute answer. Just be up front about what you're assuming, and don't try to evade homework and exam problems by declaring everything an axiom!

这让我们得以快速开始，但有时你也会认为公理的这种不精确的规范会让人苦恼。例如，你可能在证明时陷入疑惑，“我是否必须证明这一小处事实，还是可以把它当公理？”不要羞于寻求指导，但的确不存在绝对正确的答案。只需直面你的职责，不要把一切都声明为公理来逃避作业和考试中的问题！

2.1.2 Logical Deductions 逻辑推论

Logical deductions or **inference rules** are used to prove new propositions using previously proved ones.

使用之前证实的命题来证明一个新命题时，会用到**逻辑推论**或**推理规则**。

A fundamental inference rule is *modus ponens*. This rule says that a proof of P together with a proof that P *IMPLIES* Q is a proof of Q.

假言推理是一个基本的**推理规则**。该规则称，**P**为真且**P 蕴涵 Q**为真，即可证明**Q**为真。

Inference rules are sometimes written in a funny notation. For example, *modus ponens* is written:

推理规则有时用古怪的符号标记。例如，假言推理标记如下：

Rule 2.1.1

$$\frac{P,\ P\ \ IMPLIES\ \ Q}{Q}$$

规则 2.1.1

$$\frac{P,\ P\ \ \textbf{蕴涵}\ Q}{Q}$$

When the statements above the line, called the antecedents, are proved, then we can consider the statement below the line, called the conclusion or **consequent**, to also be proved.

直线上面的陈述叫做**前件**，下面的陈述叫做**结论**或**后件**。证明了前件，就可以认为也证明了后件。

A key requirement of an **inference rule** is that it must be *sound*: any assignment of truth values that makes all the antecedents true must also make the consequent true. So if we start off with true axioms and apply sound inference rules, everything we prove will also be true.

一定得合理是推理规则的必要条件：指定任何值使所有前件为真，必然使后件也为真。所以如果我们从正确的公理着手，应用合理的推理规则，那么我们所证明的一切也都将为真。

You can see why modus ponens is a sound inference rule by checking the truth table of P *IMPLIES* Q. There is only one case where P and P IMPLIES Q are both true, and in that case Q is also true.

通过核查"P 蕴涵 Q"的真值表，你就会明白为什么要说假言推理是合理的推理规则。只在一种情形下**P**与**P 蕴涵 Q**都为真，同时**Q**也为真。

P	Q	$P \to Q$
F	F	T
F	T	T
T	F	F
T	T	T

There are many other natural, sound inference rules, for example:

也有很多自然合理的推理规则，如：

Rule 2.1.2

$$\frac{P\ IMPLIES\ Q,\ Q\ \ IMPLIES\ \ R}{P\ IMPLIES\ R}$$

规则 2.1.2

$$\frac{P \text{ 蕴涵 } Q,\ Q \text{ 蕴涵 } R}{P \text{ 蕴涵 } R}$$

Rule 2.1.3

$$\frac{P\ IMPLIES\ Q,\ \ NOT(Q)}{NOT(P)}$$

规则 2.1.3

$$\frac{P \text{ 蕴涵 } Q,\ \text{非} Q}{\text{非} P}$$

Rule 2.1.4

$$\frac{NOT(P)\ IMPLIES\ NOT(Q)}{Q\ \ IMPLIES\ \ P}$$

规则 2.1.4

$$\frac{\text{非} P \text{ 蕴涵 非} Q}{Q \text{ 蕴涵 } P}$$

On the other hand,

Non-Rule.

$$\frac{NOT(P)\ IMPLIES\ NOT(Q)}{P\ \ IMPLIES\ \ Q}$$

is not sound: if P is assigned **T** and Q is assigned **F**, then the antecedent is true and the consequent is not.

另一方面，

非规则

$$\frac{非P \text{ 蕴涵 } 非Q}{P \text{ 蕴涵 } Q}$$

并不合理：如果指定 P 为**真**，Q 为**假**，那么前件为真，但后件却不为真。

Note that a propositional inference rule is sound precisely when the conjunction(AND) of all its antecedents implies its consequent.

请注意，**命题推理规则**所有前件的合取(AND)表明它的后件为真时，才可以说**该规则**真正合理。

As with axioms, we will not be too formal about the set of legal inference rules. Each step in a proof should be clear and "logical"; in particular, you should state what previously proved facts are used to derive each new conclusion.

就像对待公理那样，我们不要求太过规范的**合法推理规则集**。证明中的每一步都应清楚、“合乎逻辑”；你尤其应该声明使用了之前证实过的哪些事实来推导每个新结论。

2.1.3 Proof Templates 证明模板

In principle, a proof can be any sequence of logical deductions from axioms and previously proved statements that concludes with the proposition in question. This freedom in constructing a proof can seem overwhelming at first. How do you even start a proof?

从理论上讲，证明可以是任意的逻辑推论序列——从公理与之前证实过的陈述开始，以所讨论的命题作为结束。**证明构造**中的这种自由乍看似乎让人无所适从。究竟怎样开始一段证明?

Here’s the good news: many proofs follow one of a handful of standard templates. Each proof has it own details, of course, but these templates at least provide you with an outline to fill in. In the remainder of this chapter, we’ll go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level

outline while others help you at the next level of detail. And we'll show you other, more sophisticated proof techniques in Chapter 3.

好消息是，许多证明都遵循少数几个标准模板中的某一个。当然，每个证明都有它自己的细节，但这些模板至少为你提供了一个可以增补的框架。本章的剩余部分，我们将通读几个这类标准方法，指出基本思想、常见陷阱，并提供一些示例。许多这类模板可以组合使用；某个模板可能为你提供顶层框架，而其他模板会在另一个细节层次上帮到你。在第 3 章中，我们还将向你介绍其他更高超的证明技巧。

The recipes that follow are very specific at times, telling you exactly which words to write down on your piece of paper. You're certainly free to say things your own way instead; we're just giving you something you *could* say so that you're never at a complete loss.

下面列出的方法有时非常具体，精确地告诉你在纸上写下哪个字。与之相反，你当然可不受限制地用自己的方式来完成证明；我们只是给你一些建议，以免你全然不知所措。

2.2 Proof by Cases 分情况证明

Breaking a complicated proof into cases and proving each case separately is a useful and common proof strategy. In fact, we have already implicitly used this strategy when we used truth tables to show that certain propositions were true or valid. For example, in section 1.1.5, we showed that an implication P IMPLIES Q is equivalent to its contrapositive NOT(Q) IMPLIES NOT(P) by considering all 4 possible assignments of T or F to P and Q. In each of the four cases, we showed that P IMPLIES Q is true if and only if NOT (Q) IMPLIES NOT (P) is true. For example, if P = T and Q = F, then both P IMPLIES Q and NOT (Q) IMPLIES NOT (P) are false, thereby establishing that (P IMPLIES Q) IFF (NOT (Q) IMPLIES NOT (P)) is true for this case. If a proposition is true in every possible case, then it is true.

把复杂的证明分解为若干情况，并分别证明各个情况，这是一种又有效又常见的证明策略。事实上，我们使用真值表证明某些命题为真（或合理）时，就隐式地用过该策略。例如，在 1.1.5 小节中，通过把 T 或 F 赋给 P 及 Q 的 4 种可能情形，我们证明了蕴涵**“P 蕴涵 Q”**等价于其对换命题**“(非 Q)蕴涵(非 P)"**。在四种情形中, 我们都证明了**“P 蕴涵 Q”**为真当且仅当**“非 Q 蕴涵非 P”**为真。例如，如果 P=T,Q=F, 那么**“P 蕴涵 Q ”**和**”(非 Q)蕴涵(非 P)“** 都为假，由此证实在这种情形下，**“P 蕴涵 Q” 当且仅当“(非 Q)蕴涵(非 P)”**为真。如果某个命题在所有可能情形下都为真，那么该命题为真。

Proof by cases works in much more general environments than propositions involving Boolean variables. In what follows, we will use this approach to prove a simple fact about acquaintances. As background, we will assume that for any pair of people, either they have met or not. If every pair of people in a group has met, we'll call the group a *club*. If every pair of people in a group has not met, we'll call it a group of *strangers*.

与涉及布尔变量的命题相比，**分情况证明**在更一般的环境下有效。接下来，我们将用这一方法证明熟人关系的简单事实。作为背景，我们假定对于任何一对人，他们要么见过面，要么没见过。如果某组里每对人都见过面，我们称该组为**俱乐部**。如果某组里的每对人都没见过面，我们称它为**由陌生人构成的组**。

Theorem. *Every collection of 6 people includes a club of 3 people or a group of 3 strangers.*

定理：**每个 6 人组**都包括**一个 3 人俱乐部**或一个**由 3 个陌生人构成的组**。

Proof. The proof is by **case analysis**. Let x denote one of the six people. There are two cases:

1. Among the other 5 people besides x, at least 3 have met x.
2. Among the other 5 people, at least 3 have not met x.

证明：通过**情况分解**来证明。用 x 代表 6 个人中的某一个。有两种情况：

1. 除 x 以外的其他 5 个人中，至少有 3 个人见过 x。
2. 其他 5 个人中，至少有 3 个人没见过 x。

Now we have to be sure that at least one of these two cases must hold, but that's easy: we've split the 5 people into two groups, those who have shaken hands with x and those who have not, so one of the groups must have at least half the people.

现在我们必须确定，这两种情况中至少有一种肯定成立，这并不难：我们已经把这 5 个人分成了两组，一组和 x 握过手，一组没握过，所以其中一组必定至少有一半人。

- **Case 1:** Suppose that at least 3 people have met x.

  This case splits into two subcases:

  - **Case 1.1:** Among the people who have met x, none have met each other. Then the people who have met x are a group of at least 3 strangers. So the Theorem holds in this subcase.
  - **Case 1.2**: Among the people who have met x, some pair have met each other. Then that pair, together with x, form a club of 3 people. So the Theorem holds in this subcase.

This implies that the Theorem holds in Case 1.

- **情况 1**：假设至少有 3 个人**见过**x。

  该情况分为两种子情况：

  - **情况 1.1**：**见过 x 的人**互相都未见过面。那么**见过 x 的人**就构成了一个**至少包含 3 个陌生人的组**。所以该定理在这一子情况中成立。
  - **情况 1.2**: 见过 x 的人中，有一对彼此见过面。那么这一对和 x 一起，就构成 了**3 人俱乐部**。所以该定理在这一子情况中成立。

这表明该定理在情况 1 中成立。

- **Case 2**: Suppose that at least 3 people have not met x.

  This case also splits into two subcases:

  - **Case 2.1**: Among the people who have not met x, every pair has met each other. Then the people who have not met x are a club of at least 3 people. So the Theorem holds in this subcase.
  - **Case 2.2**: Among the people who have not met x, some pair have

not met each other. Then that pair, together with x, form a group of at least 3 strangers. So the Theorem holds in this subcase.

This implies that the Theorem also holds in Case 2, and therefore holds in all cases.■

- **情况 2**：假设至少有 3 个人**没见过**x。

  这一情况也分为两种子情况：

  - **情况 2.1**：**没见过 x 的人**，每对都互相见过面。则那些没见过 x 的人，构成了一个至少包含 3 个人的俱乐部。 所以该定理在这一子情况中成立。
  - **情况 2.2**：**没见过 x 的人**中，有一对彼此未见过面。那么这一对和 x 一起，就构成了一组至少包含 3 个陌生人的组。所以该定理在这一子情况中成立。

这表明该定理在情况 2 中也成立，因此该定理在所有情况中都成立。■

2.3 Proving an Implication 证明蕴含

Propositions of the form "If P, then Q" are called implications. This implication is often rephrased as "P IMPLIES Q" or "$P \to Q$".

有着“**如果 P, 那么 Q**”形式的命题称为蕴涵。该蕴涵常重新表述为"P 蕴涵 Q" 或 "$P \to Q$"。

Here are some examples of implications:

下面是一些蕴涵示例：

- (Quadratic Formula) If $ax^2 + bx + c = 0$ and $a \neq 0$, then

  $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- (二次公式）如果$ax^2 + bx + c = 0$ 且 $a \neq 0$, 那么

  $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- (Goldbach's Conjecture) If n is an even integer greater than 2, then n is a

  sum of two primes.
- (哥德巴赫猜想）如果 n 是一个大于 2 的偶数，那么 n 是两个素数的和。
- If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.
- 如果 $0 \leq x \leq 2$, 那么 $-x^3 + 4x + 1 > 0$.

There are a couple of standard methods for proving an implication.

有几个标准方法可以证明蕴涵。

2.3.1 Method #1: Assume P is true 方法 1：假设 P 为真

When proving P IMPLIES Q, there are two cases to consider: P is true and P is false. The case when P is false is easy since, by definition, **F** IMPLIES Q is true no matter what Q is. This case is so easy that we usually just forget about it and start right off by assuming that P is true when proving an implication, since this is the only case that is interesting.

证明**"P 蕴涵 Q"**时要考虑两种情况：P 为真及 P 为假。**"P 为假"**的情况容易证明，因为根据定义，无论 Q 是什么，"**F**蕴涵 Q"总为真。这种情况太容易了，证明蕴涵时，常常只用忽略它，通过假定"P 为真"（因为它是值得关注的唯一情形），立刻开始证明。

Hence, in order to prove that P IMPLIES Q:

1. Write, "Assume P".
2. Show that Q logically follows.

因此，为证明"P 蕴涵 Q"，得这样做：

1. 写下"假设 P 为真"。
2. 证明在逻辑上可推断出 Q。

For example, we will use this method to prove

例如，我们将使用该方法证明

Theorem 2.3.1. If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.

定理 2.3.1. 如果 $0 \leq x \leq 2$, 那么 $-x^3 + 4x + 1 > 0$.

Before we write a proof of this theorem, we have to do some scratchwork to figure out why it is true.

在写下该定理的证明前，不得不做一些草拟的工作来弄清它为真的原因。

The inequality certainly holds for x=0; then the left side is equal to 1 and $1 > 0$. As x grows, the 4x term (which is positive) initially seems to have greater magnitude than $-x^3$ (which is negative). For example, when x = 1, we have $4x = 4$, but $-x^3 = -1$. In fact, it looks like $-x^3$ doesn't begin to dominate $4x$ until $x > 2$. So it seems the $-x^3 + 4x$ part should be nonnegative for all x between 0 and 2, which would imply that $-x^3 + 4x + 1$ is positive.

x=0 时该不等式肯定成立；此时左侧等于 1，而$1 > 0$。当 x 增大时，起初项 4x(值为正）的绝对值**似乎**要大过 $-x^3$(值为负)。例如，当 x=1 时，有 $4x = 4$, 但 $-x^3 = -1$. 事实上，很可能直到$x > 2$， $-x^3$ 才开始超过 4x。所以对于 0 到 2 之间的所有 x, **好像** $-x^3 + 4x$ 这部分应该是非负值，这意味着 $-x^3 + 4x + 1$是正值。

So far, so good. But we still have to replace all those "seems like" phrases with solid, logical arguments. We can get a better handle on the critical $-x^3 + 4x$ part by factoring it, which is not too hard:

$$-x^3 + 4x = x(2 - x)(2 + x)$$

到目前为止还不错。但我们还需要把所有这些“似乎”词组替换为逻辑上的确切论据。通过把关键部分$-x^3 + 4x$分解成因子（这并不太难），我们可以更好地理解它：

$$-x^3 + 4x = x(2 - x)(2 + x)$$

Aha! For x between 0 and 2, all of the terms on the right side are nonnegative. And a product of nonnegative terms is also nonnegative. Let's organize this blizzard of observations into a clean proof.

啊哈！对于 0 到 2 之间的 x，右侧的所有项都是非负的。而非负项的乘积也是非负的。让我们把这一大堆观察所得整理成清晰准确的证明。

Proof. Assume $0 \leq x \leq 2$. Then x, 2-x, and 2+x are all nonnegative. Therefore, the product of these terms is also nonnegative. Adding 1 to this product gives a positive number, so:

$$x(2-x)(2+x)+1>0$$

Multiplying out on the left side proves that

$$-x^3+4x+1>0$$

as claimed. ■

证明：假设 $0 \leq x \leq 2$，那么 x, 2-x, 2+x 都是非负的。因此，这些项的乘积也是非负的。把 1 加到该乘积上得到一个正数，所以：

$$x(2-x)(2+x)+1>0$$

把左侧的乘积展开，就证明了前面声称的

$$-x^3+4x+1>0$$

There are a couple points here that apply to all proofs:

有两个细节适用于所有证明：

- You'll often need to do some scratchwork while you're trying to figure out the logical steps of a proof. Your sratchwork can be as disorganized as you like——full of dead-ends, strange diagrams, obscene words, whatever.But keep your scratchwork separate from your final proof, which should be clear and concise.
- 在你尝试着搞清楚证明的逻辑步骤时，常常需要做一些草拟的工作。你的草拟工作可以随心所欲——充斥着毫无进展可能的工作、奇怪的图表、令人憎恶的词等等。但要把你的草稿和最终的证明分开，后者应该清楚简明。
- Proofs typically begin with the word "Proof" and end with some sort of doohickey like □ or ■ or "q.e.d.". The only purpose for these conventions is to clarify where proofs begin and end.

- 证明通常用“证明”这个词开始，用某些像□ 、 ■ 或 "q.e.d."一样的小玩意结束。这些约定的唯一目的是阐明证明从哪开始，到哪结束。

Potential Pitfall 潜在的陷阱

For the purpose of proving an implication P IMPLIES Q, it's OK, and typical, to begin by assuming P. But when the proof is over, it's no longer OK to assume that P holds! For example, Theorem 2.3.1 has the form "if P, then Q" with P being "$0 \leq x \leq 2$ " and Q being "$-x^3 + 4x + 1 > 0$", and its proof began by assuming that $0 \leq x \leq 2$. But of course this assumption does not always hold. Indeed, if you were going to prove another result using the variable x, it could be disastrous to have a step where you assume that $0 \leq x \leq 2$ just because you assumed it as part of the proof of Theorem 2.3.1.

为了证明蕴涵“P 蕴涵 Q”这一目的，用“假设 P 为真”开始通常没问题。但证明结束后，再假设 P 成立就不行了。例如，定理 2.3.1 有着”若 P,则 Q“的格式，P 是"$0 \leq x \leq 2$ ",Q 是"$-x^3 + 4x + 1 > 0$"，证明从假设$0 \leq x \leq 2$开始。但显而易见，这一假设并非一直成立。确切来说，如果你要用变量 x 证明其他结果，有这样一个步骤——只是因为你把它看作定理 2.3.1 的一部分，就假设 $0 \leq x \leq 2$——可能会很糟糕。

2.3.2 Method #2: Prove the Contrapositive 方法 2：证明对换命题

We have already seen that an implication "P IMPLIES Q" is logically equivalent to its contrapositive

NOT(Q) IMPLIES NOT(P)

我们已经知道蕴涵”P 蕴涵 Q“在逻辑上等价于其对换命题

”非 Q 蕴涵非 P“

Proving one is as good as proving the other, and proving the contrapositive is sometimes easier than proving the original statement.

证明一个就相当于证明了另一个，而证明对换命题有时要比证明最初的陈述更为容易。

Hence, you can proceed as follows:

1. Write, "We prove the contrapositive:" and then state the contrapositive.
2. Proceed as in Method #1.

因此，你可以按照如下方式证明：

1.写下，”证明对换命题：“，然后陈述对换命题。

2.像方法 1 一样继续证明。

For example, we can use this approach to prove

Theorem 2.3.2. If r is irrational, then $\sqrt{r}$ is also irrational.

例如，我们可以用该方法来证明

定理 2.3.2. 如果 r 是无理数，那么$\sqrt{r}$也是无理数。

Recall that rational numbers are equal to a ratio of integers and irrational numbers are not. So we must show that if r is *not* a ratio of integers, then $\sqrt{r}$ is also *not* a ratio of integers. That's pretty convoluted! We can eliminate both *not*'s and make the proof straightforward by considering the contrapositive instead.

回想一下，有理数等于整数之比，而无理数却不是这样。所以我们必须证明，如果 r**不是**整数之比，那么$\sqrt{r}$也**不是**整数之比。这太让人费解了！相反，通过考虑对换命题，我们可以把两个“**不是**”都给剔除掉，使该证明更为简洁。

Proof. We prove the contrapositive:if $\sqrt{r}$ is rational, then r is rational.

Assume that $\sqrt{r}$ is rational. Then there exist integers a and b such that:

$$\sqrt{r} = \frac{a}{b}$$

Squaring both sides gives:

$$r = \frac{a^2}{b^2}$$

Since a^2 and b^2 are integers, r is also rational. ■

证明：证明对换命题：如果$\sqrt{r}$是有理数，那么 r 是有理数。

假设$\sqrt{r}$是有理数。则存在整数 a、b,使得：

$$\sqrt{r} = \frac{a}{b}$$

将两边都平方，可得：

$$r = \frac{a^2}{b^2}$$

因为 a^2 和 b^2 是整数，所以 r 也是有理数。 ■

2.4 Proving an “If and Only If” 证明“当且仅当”

Many mathematical theorems assert that two statements are logically equivalent; that is, one holds if and only if the other does.

许多数学定理断言两个陈述逻辑上等价；即，一个陈述成立**当且仅当**另一个陈述也成立。

Here is an example that has been known for several thousand years:

Two triangles have the same side lengths if and only if two side lengths and the angle between those sides are the same in each triangle.

下面是一个数千年来公认的例子：

当且仅当两个三角形的两边及它们之间的夹角相等时，这两个三角形才全等。

The phrase "if and only if" comes up so often that it is often abbreviated "iff".

我们会非常频繁地提到词组“当且仅当”，所以常把它简写为**“iff"**。

2.4.1 Method #1: Prove Each Statement Implies the Other 方法 1：证明每个陈述蕴涵另一个陈述

The statement "P IFF Q" is equivalent to the two statements "P IMPLIES Q" and "Q IMPLIES P". So you can prove an "iff" by proving two implications:

1.Write, "We prove P implies Q and vice-versa".

2.Write, "First, we show P implies Q". Do this by one of the methods in Section 2.3.

3.Write,"Now, we show Q implies P". Again, do this by one of the methods in Section 2.3.

陈述“P**当且仅当**Q"等价于两个陈述”P 蕴涵 Q“和“Q 蕴涵 P”。所以你可以通过证明两个蕴涵来证明”当且仅当“：

1.写下，”证明 P 蕴涵 Q,反之亦然”。

2.写下，“首先证明 P 蕴涵 Q”。用 2.3 节中的一种方法来证明。

3.写下，“现在证明 Q 蕴涵 P”。再次使用 2.3 节里的一种方法证明。

2.4.2 Method #2: Construct a Chain of IFFs 方法 2：构造“当且仅当”链

In order to prove that P is true iff Q is true:

1. Write, "We construct a chain of if-and-only-if implications".
2. Prove P is equivalent to a second statement which is equivalent to a third statement and so forth until you reach Q.

为证明当且仅当 Q 为真时，P 才为真，得这样做：

1.写下，“构造一个当且仅当蕴涵链”。

2.证明 P 等价于第二个陈述，第二个陈述等价于第三个陈述，依此类推，直到到达 Q 为止。

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof, as we see in the following example.

与第一种方法相比，该方法有时需要更多技巧，但正如我们在下面的示例中所看到的那样，结果却可能是个简洁优雅的证明。

Theorem 2.4.1. The standard deviation of a sequence of values $x_1,...,x_n$ is zero iff all the values are equal to the mean.

定理 2.4.1. 当且仅当序列 $x_1,...,x_n$的所有值都等于均值时，该序列的标准差才为零。

Definition. The standard deviation of a sequence of values $x_1,x_2,...,x_n$ is defined to be:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+...+(x_n-\mu)^2}{n}} \quad (2.1)$$

where μ is the mean of the values:

$$\mu ::= \frac{x_1+x_2+...+x_n}{n}$$

定义 定义序列 $x_1,...,x_n$标准差为：

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+...+(x_n-\mu)^2}{n}} \quad (2.1)$$

这里的μ是所有值的均值：

$$\mu ::= \frac{x_1+x_2+...+x_n}{n}$$

As an example, Theorem 2.4.1 says that the standard deviation of test scores is zero if and only if everyone scored exactly the class average. (We will talk a lot more about means and standard deviations in Part IV of the book.)

举个例子，按照定理 2.4.1 的理论，当且仅当每个人的得分都刚好等于班级平均分时，测验得分的标准差才为零。（在本书的第四部分，我们会更多地谈到均值与标准差）。

Proof. We construct a chain of "iff" implications, starting with the statement that the standard deviation (2.1) is zero:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+...+(x_n-\mu)^2}{n}} = 0 \quad (2.2)$$

Since zero is the only number whose square root is zero, equation(2.2) holds iff

$$(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2 = 0 \text{ (2.3)}$$

Squares of real numbers are always nonnegative, and so every term on the left hand side of equation (2.3) is nonnegative. This means that (2.3) holds iff

Every term on the left hand side of (2.3) is zero. (2.4)

But a term $(x_i - \mu)^2$ is zero iff $x_i = \mu$, so (2.4) is true iff

Every x_i equals the mean. ■

证明：我们构造了一个“当且仅当”蕴涵链，它从标准差（2.1）为零这个陈述开始：

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\ldots+(x_n-\mu)^2}{n}} = 0 \text{ (2.2)}$$

因为零是唯一一个平方根为零的数，所以**当且仅当**

$$(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_n - \mu)^2 = 0 \text{ (2.3)}$$

等式（2.2）才成立。

实数的平方总是非负的，所以等式（2.3）左手侧的每一项都是非负的。这意味着**当且仅当**

等式（2.3）左手侧的每一项都为零时 （2.4）

等式（2.3）成立。

但**当且仅当**$x_i = \mu$时，某一项 $(x_i - \mu)^2$才为零，所以**当且仅当**

每个 x_i都等于均值时，

等式（2.4）才为真。 ■

2.5 Proof by Contradiction 反证法

In a proof by contradiction or indirect proof, you show that if a proposition were false, then some false fact would be true. Since a false fact can't be true, the proposition had better not be false. That is, the proposition really must be true.

在反证法或间接证明中，证明了如果命题为假，那么某个错误的事实就会为真。因为错误的事实不能为真，所以该命题最好不要为假。也就是说，该命题必定为真。

Proof by contradiction is always a viable approach. However, as the name suggests, indirect proofs can be a little convoluted. So direct proofs are generally preferable as a matter of clarity.

反证法始终是个可行的方法。然而正如名字所示，间接证明可能会有些令人费解。所以要让证明清晰易懂，直接证明通常更合适些。

Method: In order to prove a proposition P by contradiction:

1. Write, "We use proof by contradiction."
2. Write, "Suppose P is false."
3. Deduce something known to be false (a logical contradiction).
4. Write, "This is a contradiction. Therefore, P must be true."

方法：为了用反证法证明命题 P, 得这么做：

1. 写下“使用反证法证明”。
2. 写下“假设 P 为假”。
3. 推断出已知事实为假（逻辑矛盾）。
4. 写下“导出矛盾，因此，P 肯定为真。”

As an example, we will use proof by contradiction to prove that $\sqrt{2}$ is irrational. Recall that a number is rational if it is equal to a ratio of integers. For example, $3.5 = 7/2$ and $0.1111... = 1/9$ are rational numbers.

举个例子，我们将使用反证法来证明 $\sqrt{2}$是无理数。回想一下，如果某个数等于整数之比，那么该数为有理数。如，$3.5 = 7/2$ 和 $0.1111... = 1/9$是有理数。

Theorem 2.5.1. $\sqrt{2}$ is irrational.

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d where n and d are positive integers. Furthermore, let's take n and d so that n/d is in lowest terms (that is, so that there is no number greater than 1 that divides both n and d).

Squaring both sides gives $2 = n^2/d^2$ and so $2d^2 = n^2$. This implies that n is a multiple of 2. Therefore n^2 must be a multiple of 4. But since $2d^2 = n^2$, we know $2d^2$ is a multiple of 4 and so d^2 is a multiple of 2. This implies that d is a multiple of 2.

So the numerator and denominator have 2 as a common factor, which contradicts the fact that n/d is in lowest terms. So $\sqrt{2}$ must be irrational. ■

定理 2.5.1. $\sqrt{2}$是无理数。

证明：使用反证法证明 。假设这个待证明的论点为假；也就是说，$\sqrt{2}$ 是有理数。那么就可以把 $\sqrt{2}$ 写作分数 n/d，这里的 n 和 d 都是正整数。更进一步，取 n 和 d,使得 n/d是最简分式（即，使得 n 和 d 没有大于 1 的公因子）。

把两边都平方可得$2 = n^2/d^2$，所以$2d^2 = n^2$。这间接说明 n 是 2 的倍数。因此n^2一定是 4 的倍数。又因为 $2d^2 = n^2$，可知 $2d^2$ 是 4 的倍数，所以d^2是 2 的倍数。这间接说明 d 是 2 的倍数。

所以分子和分母有个公因子 2，这与n/d 是最简分式的事实相矛盾。所以$\sqrt{2}$必定是无理数。■

Potential Pitfall 潜在的陷阱

A proof of a proposition P by contradiction is really the same as proving the implication **T** IMPLIES P by contrapositive. Indeed, the contrapositive of **T** IMPLIES P is NOT(P) IMPLIES **F**. As we saw in Section 2.3.2, such a proof would be begin by assuming NOT(P) in an effort to derive a falsehood, just as you do in a proof by contradiction.

用反证法证明命题 P 实际上和用对换命题证明蕴涵”**T**蕴涵 P“是一样的。其实，”**T**蕴涵 P“的对换命题是“非 P 蕴涵**F**"。正如我们在 2.3.2 小节所学到的，这种证明可能会以假设”非 P“为真开始，试图导出错误，正好和用反证法证明时的做法相同。

No matter how you think about it, it is important to remember that when you start by assuming NOT(P), you will derive conclusions along the way that are not necessarily true. (Indeed, the whole point of the method is to derive a falsehood.) This means that you cannot rely on intermediate results after a proof by contradiction is completed (for example, that n is even after the proof of Theorem 2.5.1). There was not much risk of that happening in the proof of Theorem 2.5.1, but when you are doing more complicated proofs that build up from several lemmas, some of which utilize a proof by contradiction, it will be important to keep track of which propositions only follow from a (false) assumption in a proof by contradiction.

无论你怎样理解，重要的是要记住，从假设”非 P“为真开始，将沿着未必正确的道路，推导出结论。(确切说来，该方法的目标是导出错误。）这意味着用反证法证明完毕后，你不能依靠它的中间结果。（如，在证明完定理 2.5.1.后，不能使用 n 是偶数这个中间结果）在证明定理 2.5.1.时，不太可能发生这种情况。但有时你会进行更为复杂的证明，它们从几个引理系统地发展而来，其中有些引理使用了反证法。这时，记录哪些命题只是从反证法中的错误假设推断出来的，就变得重要起来。

2.6 Proofs about Sets 集合的证明

Sets are simple, flexible, and everywhere. You will find some set mentioned in nearly every section of this text.

集合简单、灵活、无处不在。你会发现在本书的几乎所有章节都提到了一些集合。

In fact, we have already talked about a lot of sets: the set of integers, the set of real numbers, and the set of positive even numbers, to name a few.

In this section, we'll see how to prove basic facts about sets. We'll start with some definitions just to make sure that you know the terminology and that you are comfortable working with sets.

2.6.1 Definitions

Informally, a set is a bunch of objects, which are called the elements of the set. The elements of a set can be just about anything: numbers, points in space, or even other sets. The conventional way to write down a set is to list the elements inside curly-braces. For example, here are some sets:

$A = \{Alex, Tippy, Shells, Shadow\}$ dead pets

$B = \{red, blue, yellow\}$ primary colors

$C = \{\{a, b\}, \{a, c\}, \{b, c\}\}$ a set of sets

This works fine for small finite sets. Other sets might be defined by indicating how to generate a list of them:

$D = \{1, 2, 4, 8, 16, \ldots\}$ the powers of 2

The order of elements is not significant, so $\{x, y\}$ and $\{y, x\}$ are the same set written two different ways. Also, any object is, or is not, an element of a given set——there is no notion of an element appearing more than once in a set. So writing $\{x, x\}$ is just indicating the same thing twice, namely, that x is in the set. In particular, $\{x, x\}$=$\{x\}$.

The expression $e \in S$ asserts that e is an element of set S. For example, $32 \in D$ and blue $\in B$, but Tailspin $\notin A$ ——yet.

Some Popular Sets

Mathematicians have devised special symbols to represent some common sets.

symbol	set	elements
$\emptyset$	the empty set	none
$\mathbb{N}$	nonnegative integers	$\{0, 1, 2, 3, \dots\}$
$\mathbb{Z}$	integers	$\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$
$\mathbb{Q}$	rational numbers	$\frac{1}{2}$, $-\frac{5}{3}$, 16, etc.
$\mathbb{R}$	real numbers	π, e, -9, $\sqrt{2}$, etc.
$\mathbb{C}$	complex numbers	i, $\frac{19}{2}$, $\sqrt{2} - 2i$, etc.

A superscript "$+$" restricts a set to its positive elements; for example, $\mathbb{R}^+$ denotes the set of positive real numbers. Similarly, $\mathbb{R}^-$ denotes the set of negative reals.

Comparing and Combining Sets

The expression $S \subseteq T$ indicates that set S is a subset of set T, which means that every element of S is also an element of T (it could be that $S = T$). For example, $\mathbb{N} \subseteq \mathbb{Z}$ and $\mathbb{Q} \subseteq \mathbb{R}$ (every rational number is a real number), but $\mathbb{C} \not\subseteq \mathbb{Z}$ (not every complex number is an integer).

As a memory trick, notice that the $\subseteq$ points to the smaller set, just lik a $\leq$ sign points to the smaller number. Actually, this connection goes a little further: there is a symbol $\subset$ analogous to $<$. Thus, $S \subset T$ means that S is a subset of T, but the two are not equal. So $A \subseteq A$, but $A \not\subset A$ for every set A.

There are several ways to combine sets. Let's define a couple of sets for use in examples:

$X ::= \{1, 2, 3\}$

$Y ::= \{2, 3, 4\}$

- The union of sets X and Y (denoted $X \cup Y$) contains all elements appearing in X or Y or both. Thus, $X \cup Y = \{1, 2, 3, 4\}$.
- The intersection of X and Y (denoted $X \cap Y$) consists of all

elements that appear in both X and Y. So $X \cap Y = \{2, 3\}$.
- The set difference of X and Y (denoted $X - Y$) consists of all elements that are in X, but not in Y. Therefore, $X - Y = \{1\}$ and $Y - X = \{4\}$.

The Complement of a Set

Sometimes we are focused on a particular domain, D. Then for any subset, A, of D, we define $\overline{A}$ to be the set of all elements of D not in A. That is, $\overline{A} ::= D - A$. The set

$\overline{A}$ is called the complement of A.

For example, when the domain we're working with is the real numbers, the complement of the positive real numbers is the set of negative real numbers together with zero. That is,

$\overline{\mathbb{R}^+} = \mathbb{R}^- \cup \{0\}$.

It can be helpful to rephrase properties of sets using complements. For example, two sets, A and B, are said to be *disjoint* iff they have no elements in common, that is, $A \cap B = \emptyset$. This is the same as saying that A is a subset of the complement of B, that is, $A \subseteq \overline{B}$.

Cardinality

The *cardinality* of a set A is the number of elements in A and is denoted by $|A|$. For example,

$|\emptyset| = 0$,

$|\{1, 2, 4\}| = 3$, and

$|\mathbb{N}|$ is infinite.

The Power Set

The set of all the subsets of a set, A, is called the *power set*, $\wp(A)$, of A. So $B \in \wp(A)$ iff $B \subseteq A$. For example, the elements of $\wp(\{1, 2\})$ are $\emptyset$, $\{1\}$, $\{2\}$ and $\{1, 2\}$.

More generally, if A has n elements, then there are 2^n sets in $\wp(A)$. In other words, if A is finite, then $|\wp(A)| = 2^{|A|}$. For this reason, some authors use the notation 2^A instead of $\wp(A)$ to denote the power set of A.

Sequence

Sets provide one way to group a collection of objects. Another way is in a sequence, which is a list of objects called terms or components. Short sequences are commonly described by listing the elements between parentheses; for example, (a,b,c) is a sequence with three terms.

While both sets and sequences perform a gathering role, there are several differences.

- The elements of a set are required to be distinct, but terms in a sequence can be the same. Thus, (a,b,a) is a valid sequence of length three, but $\{a, b, a\}$ is a set with two elements——not three.
- The terms in a sequence have a specified order, but the elements of a set do not. For example, (a, b, c) and (a, c, b) are different sequences, but $\{a, b, c\}$ and $\{a, c, b\}$ are the same set.
- Texts differ on notation for the *empty sequence*; we use λ for the empty sequence and $\emptyset$ for the empty set.

Cross Products

The product operation is one link between sets and sequences. A product of sets, $S_1 \times S_2 \times \ldots \times S_n$, is a new set consisting of all sequences where the first component is drawn from S_1, the second from S_2, and so forth. For example, $\mathbb{N} \times \{a, b\}$ is the set of all pairs whose first element is a nonnegative integer and whose second element is an a or a b:

$$\mathbb{N} \times \{a, b\} = \{(0, a), (0, b), (1, a), (1, b), (2, a), (2, b), \ldots\}$$

A product of n copies of a set S is denoted S^n. For example, $\{0, 1\}^3$ is the set of all 3-bit sequences:

$$\{0,1\}^3 = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$$

2.6.2 Set Builder Notation

An important use of predicates is in set builder notation. We'll often want to talk about sets that cannot be described very well by listing the elements explicitly or by taking unions, intersections, etc., of easily-described sets. Set builder notation often comes to the rescue. The idea is to define a set using a predicate; in particular, the set consists of all values that make the predicate true. Here are some examples of set builder notation:

$$A ::= \{n \in \mathbb{N} | n \; is \; a \; prime \; and \; n = 4k+1 \; for \; some \; integer \; k\}$$

$$B ::= \{x \in \mathbb{R} | x^3 - 3x + 1 > 0\}$$

$$C ::= \{a + bi \in \mathbb{C} | a^2 + 2b^2 \leq 1\}$$

The set A consists of all nonnegative integers n for which the predicate

"n is a prime and n=4k+1 for some integer k"

is true. Thus, the smallest elements of A are:

5,13,17,29,37,41,53,57,61,73,...

Trying to indicate the set A by listing these first few elements wouldn't work very well, even after ten terms, the pattern is not obvious! Similarly, the set B consists of all real numbers x for which the predicate

$$x^3 - 3x + 1 > 0$$

is ture.In this case, an explicit description of the set B in terms of intervals would require solving a cubic equation. Finally, set C consists of all complex numbers $a + bi$ such that:

$$a^2 + 2b^2 \leq 1$$

This is an oval-shaped region around the origin in the complex plane.

2.6.3 Proving Set Equalities

Two sets are defined to be equal if they contain the same elements. That is ,$X = Y$ means that $z \in X$ if and only if $z \in Y$, for all elements, z . (This is actually the first of the ZFC axioms.) So set equalities can often be formulated and proved as "iff" theorems. For example:

Theorem 2.6.1 (*Distributive Law* for Sets). Let A, B, and C be sets. Then:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \text{ (2.5)}$$

Proof. The equality (2.5) is equivalent to the assertion that

$$z \in A \cap (B \cup C) \text{ iff } z \in (A \cap B) \cup (A \cap C) \text{ (2.6)}$$

for all z. This assertion looks very similar to the Distributive Law for AND and OR that we proved in Section 1.4(equation 1.6). Namely, if P, Q, and R are propositions, then

[P AND (Q OR R)] IFF (P AND Q) OR (P AND R)

Using this fact, we can now prove (2.6) by a chain of iff's:

III Counting 计数

Introduction 引言

Counting seems easy enough: 1, 2, 3, 4, etc. This direct approach works well for counting simple things——like your toes——and may be the only approach for extremely complicated things with no identifiable structure. However, subtler methods can help you count many things in the vast middle ground, such as:

- The number of different ways to select a dozen doughnuts when there are five varieties available.

- The number of 16-bit numbers with exactly 4 ones.

Perhaps surprisingly, but certainly not coincidentally, the number in each of these two situations is the same: 1820.

Counting is useful in computer science for several reasons:

- Determining the time and storage required to solve a computational problem——a central objective in computer science——often comes down to solving a counting problem.
- Counting is the basis of probability theory, which plays a central role in all sciences, including computer science.
- Two remarkable proof techniques, the "pigeonhole principle" and "combinatorial proof", rely on counting. These lead to a variety of interesting and useful insights.

In the next several chapters, we're going to present a lot of rules for counting. These rules are actually theorems, and we will prove some of them, but our focus won't be on the proofs per se——our objective is to teach you simple counting as a practical skill, like integration.

We begin our study of counting in Chapter 9 with a collection of rules and methods for finding closed-form expressions for commonly-occurring sums and products such as $\sum_{i=1}^{n} x^i$ and $n! = \prod_{i=1}^{n} i$. We also introduce asymptotic notations such as $\sim$, O, and Θ that are commonly used in computer science to express the how a quantity such as the running time of a program grows with the size of the input.

In Chapter 10, we show how to solve a variety of recurrences that arise in computational problems. These methods are especially useful when you need to design or analyze recursive programs.

In Chapters 11 and 12, we describe the most basic rules for determining the cardinality of a set. This material is simple yet pwerful, and it provides a great tool set for use in your future career.

We conclude in Chapter 13 with a brief digression into the final frontier of counting——infinity. We'll define what it means for a set to be countable and show you some examples of sets that are really big——bigger even than the set of real numbers.

9 Sums and Asymptotics 总和与渐近性

Sums and products arise regularly in the analysis of algorithms, financial applications, physical problems, and probabilistic systems. For example, we have already encountered the sum $1 + 2 + 4 + \ldots + N$ when counting the number of nodes in a complete binary tree with N inputs.Although such a sum can be represented compactly using the sigma notation

$$\sum_{i=0}^{log\ N} 2^i, \tag{9.1}$$

it is a lot easier and more helpful to express the sum by its closed form value

$$2N - 1.$$

总和与乘积经常出现在算法分析、财务应用、物理问题、概率系统中。例如，在计算N输入完整二进制树的节点数目时，我们已经遇到过总和 $1 + 2 + 4 + \ldots + N$。虽然可以用Σ符号简洁地表示这种总和：

$$\sum_{i=0}^{log\ N} 2^i, \tag{9.1}$$

但用它的闭合形式

$$2N - 1$$

来表示该和更为简单有用。

By *closed form*, we mean an expression that does not make use of summation or product symbols or otherwise need those handy (but sometimes troublesome) dots... Expressions in closed form are usually easier to evaluate (it doesn't get much simpler than $2N - 1$, for example) and it is usually easier to get a feel for their magnitude than expressions involving large sums and products.

“闭合形式”这个术语的意思是表达式不使用总和或乘积符号，或者采用其他方式——如，使用那些方便（但有时会带来麻烦）的点...闭合形式的表达式通常更容易求值（例如，没有比$2N-1$更简单的了），而且相比于包含大的总和与乘积的表达式，我们更容易估计闭合形式表达式的大小。

But how do you find a closed form for a sum or product? Well, it's part math and part art. And it is the subject of this chapter.

但你怎样才能发现总和或乘积的闭合形式？嗯，这既需要数学知识，也需要艺术眼光。它是本章要阐述的主题。它是本章要阐述的主题。

We will start the chapter with a motivating example involving annuities. Figuring out the value of the annuity will involve a large and nasty-looking sum. We will then describe several methods for finding closed forms for all sorts of sums, including the annuity sums. In some cases, a closed form for a sum may not exist and so we will provide a general method for finding good upper and lower bounds on the sum (which are closed form, of course).

为激发你的学习动力，本章将从涉及年金的例子开始。要弄清年金值会牵涉到大而令人生厌的总和。因此，我们将会讲几种方法，用来寻找各种总和（包括年金总额在内）的闭合形式。在某些情况下，总和的闭合形式可能不存在，因此，我们会提供一个通用方法，来找出最接近总和的上下限(它们当然是闭合形式了)。

The methods we develop for sums will also work for products since you can convert any product into a sum by taking a logarithm of the product.

我们创建的这种用在总和上的方法，也将适用于乘积，因为取乘积的对数，就可以把任何乘积转换成和。

As an example, we will use this approach to find a good closed-form approximation to

$$n! ::= 1 \cdot 2 \cdot 3 \cdots n.$$

举个例子，我们将使用该方法来找到最接近

$$n! ::= 1 \cdot 2 \cdot 3 \cdots n$$

的闭合形式。

We conclude the chapter with a discussion of asymptotic notation. Asymptotic notation is often used to bound the error terms when there is no exact closed form expression for a sum or product. It also provides a convenient way to express the growth rate or order of magnitude of a sum or product.

本章的最后面将会讨论渐近线符号。总和或乘积没有精确的闭合形式表达式时，常用渐近线符号来约束误差项。它也提供了表示总和或乘积的增长率或数量级的便捷方法。

9.1 The Value of an Annuity 年金值

Would you prefer a million dollars today or $50,000 a year for the rest of your life? On the one hand, instant gratification is nice. On the other hand, the total dollars received at $50K per year is much larger if you live long enough.

今天就给你一百万美元和余生的每一年都给你五万美元，你会选哪一个？一方面，即时满足很爽。另一方面，要是你活得够久，每年给你5万美元这种情形下的总金额要大得多。

Formally, this is a question about the value of an annuity.

An annuity is a financial instrument that pays out a fixed amount of money at the beginning of every year for some specified number of years. In particular, an n-year, m-payment annuity pays m dollars at the start of each year for n years. In some cases, n is finite, but not always. Examples include lottery payouts, student loans, and home mortgages. There are even Wall Street people who specialize in trading annuities.[1]

A key question is, "What is an annuity worth?" For example, lotteries often pay out jackpots over many years. Intuitively, $50,000 a year for 20 years ought to be worth less than a million dollars right now. If you had all the cash right away, you could invest it and begin collecting interest. But what if the choice were between $50,000 a year for 20 years and a half million dollars today? Now it is not clear which option is better.

9.1.1 The Future Value of Money

In order to answer such questions, we need to know what a dollar paid out in the future is worth today. To model this, let's assume that money can be invested at a fixed annual interest rate p. We'll assume an 8% rate[2] for the rest of the discussion.

Here is why the interest rate p matters. Ten dollars invested today at interest rate p will become $(1+p)\cdot 10 = 10.80$ dollars in a year, $(1+p)^2 \cdot 10 \approx 11.66$ dollars in two years, and so forth. Looked at another way, ten dollars paid out a year from now is only realy worth $1/(1+p)\cdot 10 \approx 9.26$ dollars today. The reason is that if we had the $9.26 today, we could invest it and would have $10.00 in a year anyway. Therefore, p determines the value of money paid out in the future.

---

[1]Such trading ultimately led to the subprime mortgage disaster in 2008-2009. We'll talk more about that in Section 19.5.3.

[2]U.S. interest rates have dropped steadily for several years, and ordinary bank deposits now earn around 1.5%. But just a few years ago the rate was 8%; this rate makes some of our examples a little more dramatic. The rate has been as high as 17% in the past thirty years.

So for an n-year, m-payment annuity, the first payment of m dollars is truly worth m dollars. But the second payment a year later is worth only $m/(1+p)$ dollars. Similarly, the third payment is worth $m/(1+p)^2$, and the n-th payment is worth only $m/(1+p)^{n-1}$. The total value, V, of the annuity is equal to the sum of the payment values. This gives: