Formalization

Readings: None.

A book consulted in the preparation of these slides states, “This book, like almost every other modern mathematics book, develops its subject matter assuming a knowledge of elementary set theory. This assumption is often not justified.”

The notation and conventions of arithmetic have been with us from childhood; those of set theory creep into mathematics in high school, and become important in university (which is why they were introduced as early as Math 135). But a gap remains between our high-level understanding of these notions and the formal systems of proof introduced in this course.

To write formulas in formal logic that express ideas from mathematical proof, we definitely have to add symbols to our language such as $+$ and $\cup$. But this is not enough, because our semantics of predicate logic allows arbitrary interpretations to be assigned to these symbols. We need a way to build in rules that restrict interpretations.

This is done by means of the idea of **axioms** introduced in the discussion of alternate systems of proof for propositional logic. Once we define a set of axioms, we can talk about the **theory** of those axioms, which is the set of all formulas provable from them. Our goal, then, is to define axioms for arithmetic and set theory.

As a short warmup, we will define axioms for group theory.

Group theory

A group is a set with one distinguished element called the identity, and two operations defined on elements of the set, one unary and one binary. We denote the identity by e, the unary operation applied to x by x^*, and the binary operation applied to x and y by $x \circ y$.

The unary operation is supposed to represent an inverse, and the binary operation is supposed to be associative. We need to say these things in the axioms if they are to hold in any theorems of the theory that follows from those axioms.

A1: $\forall x \forall y \forall z (x \circ (y \circ z) = (x \circ y) \circ z)$

A2: $\forall x (x \circ e = x)$

A3: $\forall x (x \circ x^* = e)$

Group theory is the set of all sentences ϕ such that $\Gamma \vdash \phi$, where $\Gamma = \{A1, A2, A3\}$. It is a rich and useful theory, as explored in PMath 336/346, and we will not go very far into it. In fact, we will prove just one theorem, the right-cancellation law, which states that if $x \circ z = y \circ z$, then $x = y$.

Here is a mathematical proof from the axioms.

Theorem: If $x \circ z = y \circ z$, then $x = y$.

Proof:

$x \circ z$	$=$	$y \circ z$	assumption	
$(x \circ z) \circ z^*$	$=$	$(y \circ z) \circ z^*$	substitution	
$x \circ (z \circ z^*)$	$=$	$y \circ (z \circ z^*)$	A1	
$x \circ e$	$=$	$y \circ e$	A3	
x	$=$	y	A2	$\square$

To formalize the statement of the theorem, we must recognize that the free occurrences of x, y, and z really represent implicit universal quantification.

$\phi = \forall x \forall y \forall z (x \circ z = y \circ z \rightarrow x = y)$

The mathematical proof is actually quite close to a formal proof of $\Gamma \vdash \phi$ using natural deduction. Rather than give the whole proof, we will sketch how to obtain it.

Since ϕ starts with three $\forall$ quantifiers, we must open three nested proof boxes with fresh variables x_0, y_0, z_0. Within the innermost, we must prove $x_0 \circ z_0 = y_0 \circ z_0 \rightarrow x_0 = y_0$.

To prove the implication $x_0 \circ z_0 = y_0 \circ z_0 \rightarrow x_0 = y_0$, we must open a fourth nested proof box and assume $x_0 \circ z_0 = y_0 \circ z_0$. That's the first line of the mathematical proof.

The second line is $(x_0 \circ z_0) \circ z_0^* = (y_0 \circ z_0) \circ z_0^*$. To obtain this, we use $=$i to introduce $(y_0 \circ z_0) \circ z_0^* = (y_0 \circ z_0) \circ z_0^*$, and then use $=$e to effect the substitution.

To obtain the third line, $x_0 \circ (z_0 \circ z_0^*) = y_0 \circ (z_0 \circ z_0^*)$, the mathematical proof applied A1. But A1 is a triple quantification with an implication inside. So clearly we must use $\forall$e three times to expose the implication, then use $\rightarrow$e. But the third line applied A1 on both sides of an equality, so some work similar to the second line is required.

We won't go on, because the point is clear: the mathematical proof contains all of the creativity, and sufficient detail to derive the formal proof without much thought.

A good rule of thumb for the level of detail necessary in a mathematical proof is that someone who knows nothing about the intended semantic interpretation or "content" of the proof, but who understands predicate logic and the axioms of the particular theory under consideration, should be able to derive the formal proof. In particular, all uses of axioms should be made explicit.

In practice, proofs are structured by means of intermediate lemmas and theorems, which are analogous to subroutines / functions / methods as a way of structuring code.

For practice, you might try mathematical proofs of some of the following laws of group theory, and then try converting parts of them to fragments of natural deduction proofs.

Commutative Law for Identity: $x \circ e = e \circ x$.

Uniqueness of Identity: $x \circ y = x \rightarrow y = e$.

Left Cancellation Law: $z \circ x = z \circ y \rightarrow x = y$.

Left Inverse is Right Inverse: $x^* \circ x = e$.

The axioms of arithmetic and set theory are not as easy to practice with.

There are many different models of group theory (usually called groups). For example, we could take $\mathcal{A} = \{e\}$, with $e \circ e = e$ and $e^* = e$ (for convenience, we are omitting the superscript $\mathcal{M}$). A more interesting model is $\mathcal{A} = \mathbb{Z}$, with $\circ$ interpreted as addition and * interpreted as negation.

In both these models, and many others we can think of, the commutative law $\phi = \forall x \forall y (x \circ y = y \circ x)$ is true. However, ϕ is not a theorem of group theory, as demonstrated by the next example.

If $\mathcal{A}$ consists of the following six matrices

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

with e interpreted as the first matrix, $\circ$ interpreted as matrix multiplication over the integers mod 2, and * interpreted as matrix inverse, then the commutative law does not hold.

Having demonstrated the ideas of an axiomatic approach to mathematical theory on this example, we now leave further development to PMath 336/346, and continue with the more complicated tasks of axiomatizing arithmetic and set theory.

Axioms of arithmetic

The Italian mathematician Peano, building on work of Dedekind and Grassman, suggested the following axioms to define arithmetic over the natural numbers. We first present them informally.

P1: 0 is not $n + 1$ for any n.

P2: If $n + 1 = m + 1$, then $n = m$.

P3: For any property P, if $P(0)$ is true, and $P(n)$ implies $P(n + 1)$, then $P(n)$ is true for all n.

The first two seem trivial to us, and the third is obviously the principle of induction.

Peano's axioms only need "$+1$", so we can use the constant 0 and the successor function s (with the intended meaning that $s(n) = n + 1$). This yields the following formal set of axioms:

P1: $\forall n(\neg(0 = s(n)))$.

P2: $\forall n \forall m(s(n) = s(m) \rightarrow n = m)$.

P3: $\forall P((P(0) \wedge \forall n(P(n) \rightarrow P(s(n)))) \rightarrow \forall n P(n))$

Of course, P3 is not a first-order formula; it has a quantification over a predicate. It is a second-order formula. We can replace it with a first-order axiom schema (recall that this is a way of generating axioms).

P3: For any formula ϕ with one free variable n, $(\phi[0/n] \wedge (\forall n(\phi \rightarrow \phi[s(n)/n]))) \rightarrow \forall n \phi$.

This means we have an infinite number of axioms, but this is not a problem. What *is* a problem is that, as we already know, first-order logic is not very expressive. While the second-order version is powerful enough to prove all familiar theorems of mathematics, with these three first-order axioms, we cannot even express the concept $x < y$. That is, there is no formula ϕ with two free variables x, y such that when we substitute m, n (expressed as applications of s to 0) with $m < n$, the resulting formula is a theorem.

So we add $<$ to our language and our axioms explicitly.

P1: $\forall n(\neg(0 = s(n)))$.

P2: $\forall n \forall m(s(n) = s(m) \rightarrow n = m)$.

P3: For any formula ϕ with one free variable n, $(\phi[0/n] \wedge (\forall n(\phi \rightarrow \phi[s(n)/n]))) \rightarrow \forall n \phi$.

P4: $\forall n(\neg(n < n))$.

P5: $\forall n(n < s(n))$.

P6: $\forall n \forall m \forall p(n < m \wedge m < p \rightarrow n < p)$.

But we still cannot define addition.

If we add $+$ and the constant 1, we can take away s and $<$, but we need to make sure that the additions behave as we want.

P1: $\forall n(\neg(0 = n + 1)))$.

P2: $\forall n \forall m(n + 1 = m + 1 \rightarrow n = m)$.

P3: For any formula ϕ with one free variable n, $(\phi[0/n] \wedge (\forall n(\phi \rightarrow \phi[n + 1/n]))) \rightarrow \forall n \phi$.

P4: $\forall n(n + 0 = n)$.

P5: $\forall n \forall m(n + (m + 1) = (n + m) + 1)$.

We can take away $<$ because we can define it (how?).

The new axioms governing the behaviour of addition have a computational feel. They correspond to the recursive definition of addition introduced by Grassman in 1861.

```
(define (add n m)
  (cond
    [(zero? m) n]
    [else (add1 (add n (sub1 m)))]))
```


It turns out this still is not good enough to recover what we had with our original second-order version of Peano's axioms, but it is still interesting.

This is known as Presburger arithmetic, and it is a complete, consistent, and decidable theory, as Presburger showed in 1929.

Fischer and Rabin proved in 1974 that any algorithm to decide a formula of n characters in this theory requires time at least $2^{2^{cn}}$ (for some constant c). This gives us a rare example of a task that is provably difficult (but doable). CS 365 covers other results of this form and demonstrates how to prove them.

Presburger arithmetic is not good enough because we cannot define multiplication. So we must add that as well.

P1: $\forall n(\neg(0 = n + 1))$.

P2: $\forall n \forall m(n + 1 = m + 1 \rightarrow n = m)$.

P3: For any formula ϕ with one free variable n, $(\phi[0/n] \wedge (\forall n(\phi \rightarrow \phi[n + 1/n]))) \rightarrow \forall n \phi$.

P4: $\forall n(n + 0 = n)$.

P5: $\forall n \forall m(n + (m + 1) = (n + m) + 1)$.

P6: $\forall n(n * 1 = n)$.

P7: $\forall n \forall m(n * (m + 1) = (n * m) + n)$.

These are the axioms of first-order Peano arithmetic (PA).

Using these, we can prove the associativity and commutativity of addition and multiplication, and the distributive laws.

But are they expressive enough? We can finally express formally some of the ideas from Math 135 that we discussed in the first lecture. The statement "n is an even number" is formalized as $\exists m(n = m + m)$. The statement "n is prime" is formalized as $1 < n \wedge \neg \exists m \exists p(m < n \wedge p < n \wedge n = m * p)$.

Still, you would be excused for believing that we have to add something like exponentiation next, and so on, forever. This is not the case.

In fact, any function that is computable (by a Turing machine, in the lambda calculus, or in any programming language) can be defined in PA. For any computable function f of one argument, there is a formula ϕ with two free variables x, y such that $f(m) = n$ if and only if $\phi[m/x][n/y]$ is a theorem of PA. Once again we see the idea that proofs are programs and programs are proofs.

But this power comes at a cost. Gödel proved in 1931 that if PA is consistent, then it is incomplete. He constructed a statement that was semantically true but that had no proof, by coding up formulas and proofs as numbers and then creating a formula with code n that asserted that the formula with code n had no proof. (Why must this statement be true?)

He further showed that the consistency of PA could not be proved within PA. Finally, Turing and Church's work show that PA is undecidable.

These proofs suggest that more axioms need to be added, but this doesn't help. Gödel's proofs hold for any system powerful enough to express the axioms of PA. By adding more axioms, one can create a system that can prove the consistency and completeness of PA (Gentzen proved this in 1936), but one doesn't know if that system is consistent and complete.

Gödel's incompleteness theorem is not an impediment to working mathematicians, just as undecidability is not an impediment to working computer scientists. It represents a limitation of which one must be aware.

Although exponentiation can be expressed in PA by some formula ϕ, it may not be natural to do so, and it doesn't hurt to add a symbol (say $\uparrow$) for it to the language, and an axiom formalizing "$p = m \uparrow n$ if and only if $\phi[m/x][n/y][p/z]$". Any proof in the new system can be translated into one in the old system, and vice-versa.

Mathematicians and computer scientists do this sort of extension all the time in mathematical proofs. They also resort to second-order reasoning when it is useful (think of calculus, whose fundamental theorems tend to involve quantification over functions).

The expressibility of any computable function in PA suggests that we can code representations of sets and set-theoretic functions. We can, but we will not pursue that here.

Axioms of set theory

Set theory was invented by Cantor in the 1880's and first axiomatized by Frege (who invented the idea of proof systems about the same time). Surprisingly, after adding a single predicate, namely $\in$ to represent set membership, Frege needed only two axioms to express all the set-theoretic notation with which we are familiar. The first axiom says that two sets are equal iff they contain the same elements.

S1: (Extensionality) $\forall S \forall T (x \in S \leftrightarrow x \in T) \rightarrow S = T$

Note that we are using $x \leftrightarrow y$ as a synonym for $(x \rightarrow y) \wedge (y \rightarrow x)$, and that we are using the convention of using capital letters for variables representing sets.

The second axiom says that, given a unary predicate, there is a set consisting of everything satisfying the predicate.

S2: (Comprehension) $\forall P \exists S \forall x (x \in S \leftrightarrow P(x))$

We have the same problem with S2 as with Peano's original axiom of induction. S2 is a formula of second-order logic, but we can replace it by a first-order axiom schema generating axioms from formulas with one free variable (expressing predicates).

With these two axioms, we can describe more familiar set-theoretic notation. For example, we may describe a fixed finite set as $\{x_1, x_2, \ldots, x_n\}$. The corresponding formula is:

$$\forall x (x \in S \leftrightarrow (x = x_1 \vee x = x_2 \vee \ldots \vee x = x_n))$$

We may also describe a set as $\{x \mid P(x)\}$. S2 guarantees that such a description identifies a set, and S1 guarantees that it is unique.

Math 135 also introduced the union ($\cup$) and intersection ($\cap$) operators. We can also talk about these without explicitly introducing them.

Clearly, x being in the union of sets S and T is expressed by the formula $x \in S \vee x \in T$, and x being in the intersection of S and T is expressed by $x \in S \wedge x \in T$.

The subset relation $S \subseteq T$ can be expressed as $\forall x (x \in S \rightarrow x \in T)$, and the proper subset relation $S \subsetneq T$ can be expressed as $\forall x (x \in S \rightarrow x \in T) \wedge \exists x (\neg (x \in S) \wedge x \in T)$.

This looks good; there seems to be no problem with expressiveness. But the problem comes from another direction, and it is far more serious than the problems with the early axiomatizations of arithmetic.

The axioms given so far are inconsistent. Consider the set defined by the axiom of comprehension using the predicate $P(x) = \neg (x \in x)$.

This seems absurd: surely no set contains itself? But the predicate is not forbidden by the language, and the set of all sets does contain itself.

In modern notation, if $S = \{x \mid x \notin x\}$, then when we ask the question "Is $S \in S$?", we reach a contradiction no matter what the answer is.

Frege was informed of this problem, discovered by Russell just as Frege was to publish the second volume of his master work axiomatizing all known mathematics, in 1902. The problem is clearly with the axiom of comprehension, which we now remove.

Russell used a nice story to illustrate the problem. Suppose, in a village, there is a barber who shaves everyone who does not shave themselves. The question is: who shaves the barber?

A number of solutions were proposed to resolve this problem, but the solution which was most useful was proposed by Zermelo (1904) and made more precise by Fraenkel (1922). This solution is now known as the Zermelo-Frankel axioms of set theory, or ZF.

We will give a quick and incomplete sketch of ZF. The initial idea is to avoid Russell's paradox by building up sets out of literally nothing.

We start with an axiom, P2, proclaiming the existence of a set: the empty set, ϕ. We then use the following four axioms to generate more sets.

P2: (Empty) $\exists S(\forall x \neg(x \in S))$.

Axiom P3 says that given two sets T and U, there is a set $\{T, U\}$.

P3: (Unordered pairs)
$\forall T \forall U \exists S \forall w(w \in S \leftrightarrow w = T \vee w = U)$

Axiom P4 says that if we have a set $S = \{S_1, S_2, \ldots\}$, we can form the union $\cup_i \{S_i\}$.

P4: (Union) $\forall S \exists T \forall x(x \in T \leftrightarrow \exists W(W \in S \wedge x \in W))$

P5, the new axiom of replacement, says that for a set S, if there is a formula $P(x, y)$ which defines a function on S – that is, for $x \in S$, there is a unique y such that $P(x, y)$ is true – then there is a set T containing all such y.

These axioms allow us to define lots of finite sets, but they may look strange to us. P2 gives us the empty set, ϕ. P3, applied to ϕ and ϕ, gives us $\{\phi\}$. If we do this again with $\{\phi\}$, we get $\{\{\phi\}\}$. If we apply P3 to ϕ and $\{\phi\}$, we get $\{\phi, \{\phi\}\}$. These sets can represent sets that are more familiar-looking.

We can also define union, intersection, subset, proper subset, and prove various laws such as deMorgan's laws for sets. What we cannot do yet is express the idea of an infinite set.

We add **P6**, the axiom of infinity, which says that there is a set that contains the empty set and, if it contains s, then it contains $s \cup \{s\}$. This may seem strange, but it nicely embeds arithmetic in set theory.

Starting with ϕ and applying the function $f(s) = s \cup \{s\}$, we get $\{\phi\}$. Applying it again, we get $\{\phi, \{\phi\}\}$. We can think of these as $0, 1, 2$ respectively; in fact, these sets contain $0, 1, 2$ elements.

In this fashion, we can define set analogues of the natural numbers, and set-theoretic operations corresponding to addition, multiplication, and so on. The axiom of infinity states that there is a set $\mathbb{N}$ containing all of the natural numbers. It's not hard to prove the Peano axioms for this representation.

Thus we can embed all of arithmetic inside set theory.

There are three more axioms of ZF designed to create more infinite sets, work with them, and rule out certain paradoxes: they are the power set axiom, the axiom of choice, and the axiom of regularity. These can be found in books and Web pages on set theory.

These axioms form the basis of modern mathematics. However, since arithmetic is embedded in set theory, Gödel's incompleteness theorems hold for the theory of sets as well.

We will assume both the functions and predicates of arithmetic ($+$, $*$, $<$) and of set theory ($\in$, $\cup$, $\subseteq$, etc.) and use them as needed in order to express properties in our proofs and in reasoning about our programs.

Guidelines for formalization

We are now finally ready to look more closely at the art of formalizing statements in English or a mixture of English and mathematical notation. We've already seen some of the issues involved when we made various axioms more precise.

For example, we've seen that the statements of mathematical theorems often have implicit quantifiers, but these must be made explicit in a formal representation. "An even number is not prime" needs to be translated as $\forall x(E(x) \to \neg P(x))$. If we omit the quantifier, x would be free in the formula, and the formula would then be a way of expressing a unary predicate.

We used $E(x)$ as the predicate "x is even" on the previous slide, but we know how to express this: $\exists y(x + x = y)$. We also know how to express a primality predicate, but it's more awkward. E and P are more readable.

But if we choose to use these unary predicates, we have to allow for the possibility that a model will interpret E in a completely different fashion. There are many ways to restrict that interpretation. We could add a definition-style axiom:

$$\forall x(E(x) \leftrightarrow \exists y(x + x = y))$$

or we could add a series of axioms specifying the behaviour of the predicate:

$$E(0), \quad \forall x(E(x) \leftrightarrow \neg E(x + 1)), \quad \text{etc.}$$

In what follows, we will be using letter predicates for readability, but you should understand that we're assuming restricted interpretation of those predicates.

The English word "and" usually, but not always, implies a conjunction. "x and y are integers" is translated as $I(x) \wedge I(y)$. Other English words also lead to conjunction in translations: "but", "although", "moreover", "however". But sometimes the relationship is expressed by a predicate: "u and v are joined by an edge" becomes $R(u, v)$, where R is the edge relation of the graph.

In the phrase "Primes and pseudoprimes pass the Fermat test", the word "and" does not imply a conjunction. The translation is $\forall x(P(x) \vee S(x) \to F(x))$. (We'll discuss quantification shortly.)

The word "unless" requires further interpretation. Consider "The transaction is completed, unless the system fails." If the transaction is completed, the system has not failed, and if the system does not fail, the transaction is completed. This has the form $C(t) \leftrightarrow \neg F(s)$.

On the other hand, "The packet will arrive unless there is congestion on the network" allows for the possibility that even though there is congestion, the packet could still arrive. Hence the proper translation of this is $A(p) \vee C(n)$.

In either case, we use additional information beyond the single statement being translated to resolve its meaning.

The translation $p \rightarrow q$ can come from many different sentence forms: "If p, then q"; "p implies q"; "p, therefore q"; "p, hence q"; "q if p"; and "q provided that p".

Some sentences that look as if they should involve implication may not. Examples include "q because p" and "Since p, q". Both of these suggest that both p and q are true, so the proper translation is $p \wedge q$.

Mathematical proofs sometimes talk about "necessary and sufficient conditions". Rather than memorize which is which, you can reason about the phrase.

"p is a necessary condition for q". This means that p must hold before q can hold. It doesn't mean that q is guaranteed to hold once p holds. So the translation is $q \rightarrow p$.

"p is a sufficient condition for q". To get q to hold, it suffices to have p hold. There might be other ways to get q to hold, but p definitely does it. So the translation is $p \rightarrow q$.

"p is a necessary and sufficient condition for q" translates as $(p \rightarrow q) \wedge (q \rightarrow p)$, which we have been abbreviating as $p \leftrightarrow q$. But this is also commonly phrased as "p iff q" or "p if and only if q", which is short for "p if q, and p only if q".

To understand this, let's look at the phrase "p only if q". This suggests that q holding is necessary for p to hold. Thus if p holds, it must be the case that q holds. The translation is then $p \rightarrow q$.

You may also see "p only when q", which also has the translation $p \rightarrow q$.

To study translations that involve quantification, let's start by examining variations on English sentences of the form "All A's are B's"(for example, "All prime numbers are odd"). This is typically translated as $\forall x(A(x) \rightarrow B(x))$.

We know from the quantifier equivalences discussed in the previous module that this can also be written $\forall x(\neg A(x) \vee B(x))$. This also demonstrates that an translation like $\forall x(A(x) \wedge B(x))$ is incorrect.

The sentence “No A’s are B’s” is translated $\forall x(A(x) \rightarrow \neg B(x))$. It sometimes helps to paraphrase the English, as long as you do it correctly: “If something is an A, then it is not a B.”

This suggests that the translation “$\forall x \neg (A(x) \rightarrow B(x))$” is incorrect. But why? Transform the implication and you’ll see.

Similarly, the translation “$\neg \forall x(A(x) \rightarrow B(x))$” is incorrect.

The English phrase “Some A’s are B’s” is usually translated $\exists x(A(x) \wedge B(x))$. But there is a small ambiguity in the use of the English word “some”.

Sometimes it means “at least one, maybe all”, as in the phrase “Some of you will pass CS 245”. Sometimes it means “Not all”, as in the phrase “Some of you will fail CS 245”. (In this case, “maybe none” is a possibility, but often it isn’t.)

You have to think about the phrase “Only A’s are B’s”. It allows for A’s which are not B’s, but there can’t be B’s which are not A’s. The translation is therefore $\forall x(B(x) \rightarrow A(x))$.

The pattern in the translations we have seen so far is that a universal quantification often has an implication within. Here’s an example from a network specification: “All packets that arrive contain correct routing information.” This is of the form $\forall x(A(x) \rightarrow R(x))$. The formula $\forall x(A(x) \wedge R(x))$ says something quite different.

An existential quantification, on the other hand, often has a conjunction within. “Some arriving packets cause buffer overflow” is translated as $\exists x(A(x) \wedge O(x))$.

Existential quantifications rarely have implications inside, because it is too easy to make an implication true. Consider “If a packet does not contain correct routing information, it never arrives”. This looks like an implication, but we can’t use existential quantification with it.

How about $\exists x(\neg R(x) \rightarrow \neg A(x))$ as a translation of “If a packet does not contain correct routing information, it never arrives”? That formula is made true by a packet which does have correct routing information, regardless of whether or not it arrives.

We can try $\exists x(\neg R(x) \wedge \neg A(x))$, but this requires the existence of a packet without correct routing information that also does not arrive. Why should such a packet exist if we code our system properly?

The correct translation is $\forall x(\neg R(x) \rightarrow \neg A(x))$, which is equivalent to $\forall x(R(x) \wedge \neg A(x))$.

Indefinite articles in English (“a”, “an”) are also a source of ambiguity. Sometimes they result in existential quantifiers, and sometimes in universal quantifiers.

“A prime number greater than two is even” has the translation $\forall x((P(x) \wedge (x > 2)) \rightarrow E(x))$.

But “If the system crashes, a buffer has overflowed” has the translation $C(s) \rightarrow \exists x(B(x) \wedge O(x))$.

There are times when you can share quantifiers. For example, the phrase “All strings are in Unicode, but no strings exceed length 256” has the translation $\forall x(S(x) \rightarrow U(x)) \wedge \forall x(S(x) \rightarrow \neg L(x))$. But the translation $\forall x((S(x) \rightarrow U(x)) \wedge (S(x) \rightarrow \neg L(x)))$ will also work.

The reason is not because both parts of the conjunction refer to the same type of object. “All strings are in Unicode, but no ID numbers are negative” can be translated as $\forall x((S(x) \rightarrow U(x)) \wedge (I(x) \rightarrow \neg N(x)))$.

Scope rules mean that quantifier variables can be reused: $\forall x(A(x) \rightarrow \forall x B(x))$ is perfectly legal. But $\forall x(A(x) \rightarrow \forall y B(y))$ is equivalent and more readable.

What’s next?

We now have a fairly clear idea of how to formalize both notions of mathematical proof and reasoning in English or other natural languages. We will use these ideas in applications to computer science.

First, we describe and use the software tool Alloy, which accepts specifications of systems written in a language reminiscent of first-order logic, and attempts to either discover inconsistencies or demonstrate that all small models of the specifications are inconsistent.

Next, we will study one way of applying logical ideas to reasoning about the behaviour of programs.

Goals of this module

We don’t expect you to memorize any of the sets of axioms described in this module, but you should be aware of the properties of axioms needed to describe arithmetic and set theory, and of issues surrounding those sets of axioms (consistency, completeness, decidability).

You should gain some ability to be able to translate mathematical or English reasoning into formulas of predicate logic, using additional predicates or notation from arithmetic or set theory if required. This is not a matter of memorizing rules, but of understanding the common meaning of phrases used in such reasoning, and applying common sense.