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:

$$
\begin{array}{rcll}
x \circ z & = & y \circ z & \text{assumption} \\
(x \circ z) \circ z^* & = & (y \circ z) \circ z^* & \circ z^* \text{ to both sides} \\
x \circ (z \circ z^*) & = & y \circ (z \circ z^*) & \text{A1} \\
x \circ e & = & y \circ e & \text{A3} \\
x & = & y & \text{A2}
\end{array}
$$

□

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 $=$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.

Exercise: Prove this (use compactness).

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). The next slide gives an example of their use.

1	$\forall n(\neg(0 = n + 1))$	premise (P1)
2	$\forall n(n + 0 = n)$	premise (P4)
3	$0 = 1$	assumption
4	$1 + 0 = 1$	$\forall n\mathrm{e}\,2$
5	$1 + 1 = 1$	$=\mathrm{e}\,3,\ 4,\ z,\ 1 + z = 1$
6	$0 = 1 + 1$	$=\mathrm{e}_R\,5,\ 3,\ z,\ 0 = z$
7	$\neg(0 = 1 + 1)$	$\forall n\mathrm{e}\,1$
8	$\bot$	$\neg\mathrm{e}\,7,\ 6$
9	$\neg(0 = 1)$	$\neg\mathrm{i}\,3,\ 8$

Using the Peano axioms, we can (with some effort) 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.

Gödel's incompleteness theorem

But this power comes at a cost. Up until now, the logics we have considered have been sound and complete with respect to their standard semantics. In fact, the correspondence between proof theory and semantics has seemed so natural as to almost suggest that we can blindly accept soundness and completeness as givens in any logic—that everything we could prove was true, and everything we could observe to be true was provable in the system.

In reality, we have been very lucky. The fact that our proof theories to date have been strong enough to prove any truth without proving any falsehood is truly a remarkable phenomenon—we lose this property in PA.

Gödel proved in 1931 that if PA is consistent, then it is incomplete.

In other words, either the proof theory of PA (i.e. natural deduction augmented with the first-order Peano Axioms):

- is inconsistent (i.e., there is no model that satisfies all of its axioms); or
- it is incomplete (i.e., for every model of PA, there are true statements that cannot be proved).

Since the natural numbers form a model for PA (and we believe they exist), Gödel's theorem implies that there are true statements about the natural numbers that cannot be proved in PA.

How did he do it?

We will not be able to present all of the details of Gödel's proof here, but we will outline some of the highlights.

First key observation: Every formula and every proof is simply a sequence of symbols. Using some standard encoding (e.g. ASCII or Unicode with zero-padding on the left), we can view each symbol as a number. Then the concatenation of the numbers encoding each symbol yields a (really, really big) number encoding the entire formula or proof.

We call the number constructed as above the **Gödel number** of the formula or proof.

Since formulas and proofs can both be encoded as numbers, the proof rules of PA become procedures for manipulating numbers, and the notion of provability becomes a statement about natural numbers — the statement that ψ has a proof Ψ is a statement that the Gödel number of Ψ encodes a procedure for producing the Gödel number of ψ.

Second key observation: Since provability is a machine-checkable property of the natural numbers (i.e., you could write a program to determine whether a given proof string proves a given formula), it has an encoding in PA.

Thus there exists a formula ϕ in PA with free variables u and v that is true precisely when v (encoded as a Gödel number) is a proof of u (encoded as a Gödel number).

Gödel's big step

Gödel constructed the statement

$$\chi := \neg\exists v(\phi[n/u]) \ ,$$

asserting that the formula with Gödel number n has no proof. He then arranged for χ itself to have Gödel number n.

The formula thus refers to itself, and may then be read as

I have no proof in PA.

(Why must this statement be true?)

But how do we know that such a statement exists?

Is it even possible to construct a statement that contains its own Gödel number?

Self-referential statements

A famous statement that refers to itself is the Liar Paradox:

This statement is false.

The statement asserts its own negation and is therefore contradictory whether it is assumed true or false.

The self-contradiction arises from the ability of the words “This statement” to make a statement refer to itself—English has a recursion construction, of sorts.

Note that self-reference need not be paradoxical. For example:

This sentence contains five words.

is a self-referential sentence that is semantically true.

But PA has no explicit recursion construction — there is no syntactic element in predicate logic (and by extension, PA) whose meaning is “this formula”.

So it would seem that constructing a formula that refers to itself (i.e., makes an assertion about its own Gödel number) might be difficult, if not impossible.

Self-printing programs

Consider the following problem: write a program that prints its own source code.

Programming languages generally do not contain a construction whose meaning is “my source code” (where “my” refers to the program itself). So can it be done?

The following C program prints its own source code:

```
main(a){printf(a="main(a){printf(a=%c%s%c,34,a,34);}",34,a,34);}
```

(Author: Dario Dariol; source: http://www.nyx.net/~gthompso/self_c.txt)

Strictly speaking, the `#include <stdio.h>` directive is missing, but we will overlook this point.

How does this program work?

If we examine it closely, we see that it assigns a variable `a` to a string representing a portion of the program's source, with a hole left in the middle (the `%s`). The program then arranges to print `a` twice—once normally, and once embedded inside itself (at the point where the hole occurs), surrounded by quotation marks (ASCII 34).

The characteristics we have outlined here are typical of most self-printing programs.

The self-printing C program demonstrates that is possible for a program to make reference to itself without the need for a specific “my source code” command.

Can the same be done in other contexts? Can an English sentence refer to itself without using words like “this sentence”?

Quotation marks and the use-mention distinction

The use of quotation marks in the self-printing C program is more than a syntactic triviality. Consider the following sentence:

This sentence is not a sentence.

The sentence is clearly false, as it is a perfectly good sentence.

Consider now what happens when we introduce quotation marks:

“This sentence” is not a sentence.

The sentence is now true, as the phrase “This sentence” is indeed not a sentence.

These two sentences illustrate a phenomenon known as the **use-mention distinction**. In the first case, the words “This sentence” were used to refer to the sentence itself in which they occurred. This is an example of **use**, as we are making use of the actual meaning of the words “This sentence”.

In the second case, the quoted “This sentence” refers only to the sequence of words consisting of “This”, followed by “sentence”. The meanings of these words are irrelevant, as we are only interested in the words themselves; this is an example of **mention**.

The distinction between use and mention will be familiar to Scheme programmers, as the quotation operator (’) in Scheme essentially switches the interpretation of code from a **use** mentality to a **mention** mentality.

We can play interesting games with the use-mention distinction. For example, we can use and mention the same words in the same sentence:

“Is not a sentence” is not a sentence.

Even more interesting is if those sentence fragments talk about the action itself of combining use and mention:

“Yields falsehood when preceded by its quotation” yields falsehood when preceded by its quotation.

Note that in this sentence, we are simultaneously using, mentioning, and **performing** the act of preceding a sentence fragment by its quotation.

The resulting sentence is called **Quine’s paradox**.

Quine’s Paradox

“Yields falsehood when preceded by its quotation” yields falsehood when preceded by its quotation.

But what does this sentence mean?

It refers to the sentence obtained by preceding the words “Yields falsehood when preceded by its quotation” by their quotation. If we do that, we obtain

“Yields falsehood when preceded by its quotation” yields falsehood when preceded by its quotation.

But this is exactly the original sentence!

So this sentence talks about itself, without explicitly referring to itself!

Moreover, it then asserts its own falsehood, and is therefore equivalent to the more familiar “This statement is false.”

So it is possible for an English sentence to refer to itself without using words like “this statement”.

(In a similar vein, in programming languages where functions are first-class values (like in Scheme), it is possible to build recursive functions (a form of self reference) without ever having a function explicitly call itself—see CS442.)

Can the same be done in PA, i.e., can a formula refer to its own Gödel number?

The answer is **YES!** The technique is similar to the self-printing C program.

Constructing a self-referential PA formula

Our first observation is that, given a natural number n that denotes the Gödel number of some formula, it is possible to determine those parts of n that denote free variables.

Moreover, given numbers m, n and p, it is possible to write a computer program to decide whether p is the Gödel number resulting from replacing all free variables in the Gödel number m with n.

Thus there is a formula in PA with free variables m, n, and p that encodes this notion of subsititution. Let us call it SUB. Thus,

$$\mathrm{SUB}[x/m][y/n][z/p]$$

is a theorem if and only if z is the Gödel number resulting from replacing all free variable occurrences in the Gödel number x with the number y.

Combining use and mention in PA

Now consider a step similar in spirit to combining a sentence fragment with its quotation—suppose we wish to substitute **a term's own Gödel number** for its free variables. In particular, suppose we consider evaluating the formula

$$\mathrm{SUB}[x/m][x/n][z/p] \; .$$

Then this formula is true precisely when z results from substituting the number x for all occurrences of a free variable in the Gödel number x.

To keep the notation succinct, we can define a formula AQ with free variables m and p, such that

$\mathrm{AQ}[x/m][z/p]$ is defined to mean $\mathrm{SUB}[x/m][x/n][z/p]$.

The letters AQ stand for the word **Arithmoquine**, a term coined by Hofstadter to denote the operation of substituting a formula's own Gödel number in for its free variables, in analogy to using and mentioning the same words to make up a sentence.

For example, if we have a formula ψ with free variable n that reads

$$\psi := n \text{ is prime ,}$$

then the result of arithmoquining ψ would be

$$\psi' := \text{(the Gödel number of } \psi\text{) is prime .}$$

Here, n, a free variable, is replaced with a specific number, namely the Gödel number of ψ. The result is a new formula ψ', with its own Gödel number, which is surely different from ψ's Gödel number.

The formulas ψ and ψ' would then satisfy $\mathrm{AQ}[\psi/m][\psi'/p]$.

Now, let us construct a formula ω as follows:

$$\omega := \neg\exists v \exists x(\phi[x/u] \wedge \mathrm{AQ}[z/m][x/p])$$

(Recall that ϕ has free variables u and v.) The formula ω has one free variable z, and states that the result of arithmoquining z does not have a proof.

Now, in the spirit of Quine's paradox, we arithmoquine this formula ω, which itself talks about arithmoquining:

$$\chi := \neg\exists v \exists x(\phi[x/u] \wedge \mathrm{AQ}[(\text{Gödel number of } \omega)/m][x/p])$$

Now, what is the Gödel number of χ? Since χ is obtained by arithmoquining ω (i.e. substituting ω's own Gödel number for ω's free variables), the Gödel number of χ is the result of arithmoquining the Gödel number of ω.

$$\chi := \neg\exists v \exists x(\phi[x/u] \wedge \mathrm{AQ}[(\text{Gödel number of } \omega)/m][x/p])$$

We know that χ's Gödel number is the number resulting from arithmoquining ω.

Now, what do we know about the variable x? By its use in the subformula $\mathrm{AQ}[(\text{Gödel number of } \omega)/m][x/p]$, we know that x is the Gödel number resulting from arithmoquining ω.

In other words, x is the Gödel number of χ itself—the formula χ contains an encoding of its own Gödel number!

So we read the formula χ as follows: the formula resulting from arithmoquining ω does not have a proof in PA.

But χ itself is the result of arithmoquining ω. Hence, χ reads, "χ has no proof in PA." More colloquially, χ says, "I have no proof in PA."

So we can indeed construct a formula

$$\chi := \neg\exists v(\phi[n/u]) \ ,$$

such that χ's own Gödel number is n. Effectively, as we mentioned before, this formula asserts

I have no proof in PA.

Now, is this formula χ semantically true or false? If it is false, then χ must have a proof in PA, which would mean that a false statement is provable (in which case PA would be inconsistent). If it is true, then as the statement asserts, χ is a true formula without a proof (in which case PA is incomplete).

Thus PA is either inconsistent or incomplete, and since the natural numbers form a model for PA, we conclude that PA is incomplete.

Interpreting χ

We know that the formula χ asserts its own non-provability because it contains an encoding of its own Gödel number.

But if we want, we can ignore the fact that χ's own Gödel number and the number in the formula are the same; then χ is simply a statement about a particular number, namely that that number does not denote a provable formula.

But provability is itself a high-level interpretation of a lower-level formula made entirely of quantified statements involving $+$ and $*$. So is AQ.

On a low level, then, χ is simply a number-theoretic fact, of no particular significance, that happens not to possess a proof.

But if we know χ is true....

Since we know χ to be a true statement about numbers without a proof, why not just add χ to PA as an axiom?

Then χ would become provable; the proof of χ would be as follows:

1	χ	axiom

Adding χ as an axiom should not render the system inconsistent, since we already know that χ is semantically true.

But χ asserts its own non-provability, so if χ is now provable, isn't that a contradiction?

No, it's not a contradiction. The formula χ asserts that χ is not provable **in PA**.

But the system we're working in is now PA+χ, and χ's provability in PA+χ does not contradict χ's non-provability in PA.

In fact, in the context of PA+χ, χ might not be a statement about its own non-provability anymore. In fact χ might not even be self-referential, or even a statement about provability!

At the lowest level of interpretation (i.e., quantified expressions in $+$ and $*$), χ's meaning is unchanged.

But the procedure for checking proofs in PA+χ is different from the corresponding procedure for PA, because there is a new form of proof justification available for checking (i.e., the axiom χ). Thus, the formula expressing provability in PA+χ is bound to be different from ϕ, the provability predicate for PA. Thus, χ may not be a statement about provability in the context of PA+χ.

So we can safely add χ to PA and obtain a system in which χ is provable. Does this solve all of our problems?

χ is only the beginning

Proofs in PA+χ are also machine checkable, so there exists a formula ϕ_χ expressing provability in PA+χ. We can then construct a new formula χ':

$$\chi' := \neg\exists v(\phi_\chi[n/u]) \ ,$$

such that χ''s own Gödel number is n. Then we have a formula that asserts its non-provability in PA+χ!

We could, of course, add χ' as an axiom to this system, obtaining PA+χ+χ'. But then we could construct an unprovable formula in this logic as well.

We could even envision adding the entire sequence of $\chi^{(i)}$'s to PA at once; still, we could contruct an unprovable truth.

The full incompleteness theorem

The full version of Gödel's Incompleteness Theorem expresses these difficulties:

Every *machine-checkable proof system strong enough to prove all of the theorems in PA is either inconsistent or incomplete.*

So there is **no way** to strengthen the proof system to make it complete with respect to the properties of the natural numbers.

Gödel's Second Incompleteness Theorem

The first incompleteness theorem guarantees the existence of a true statement χ that is not provable in PA. But χ itself is not likely to be a formula of interest in applications of PA.

We might be led to wonder whether there are more interesting unprovable results in PA or its extensions. Gödel's second incompleteness theorem gives a much more specific example of an unprovable statement in PA.

In particular Gödel's second incompleteness theorem states that PA cannot prove a statement of its own consistency.

As before, 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. Further, Gödel’s second incompleteness theorem says that the augmented system then cannot prove its own consistency. So again we must augment the system, and again it can prove the consistency of the smaller system, but not of itself....

More specifically, Gödel’s second incompleteness theorem states that if PA, or any extension of it with checkable proofs, can prove its own consistency, then it is inconsistent.

We speculated in Module 01 about whether a system could prove its own consistency. We decided then that even if it could, that we could not take this proof as a guarantee of its consistency—for if a system is inconsistent, then the statement of its consistency may be one of the false statements that it can prove!

Now we know that systems strong enough to represent number thoery **cannot** prove their own consistency. Thus we can never be ultimately sure about the consistency of our proof systems, or of mathematics in general. We must, therefore, eventually take on faith that our modes of reasoning are indeed consistent.

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

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.

To extensionality (now P1), we add an axiom, P2, proclaiming the existence of a set: the empty set, $\emptyset$. 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.

The power set axiom (**P7**) states that if S is a set, then the collection of all subsets of S (called the **power set** of S) is also a set:

P7: $\forall S \exists T \forall x (x \in T \leftrightarrow x \subseteq S)$.

Here T is the power set of S; the elements of T are precisely the subsets of S.

“Derived” axioms

We now exhibit an axiom that is useful in practice, but technically follows from the existing axioms. We also show that one of our existing axioms is actually a derived axiom. In order to show these results, we first formally present the axiom of replacement:

P5 $\forall S \forall P((\forall x \forall y \forall z(x \in S \wedge P(x,y) \wedge P(x,z) \rightarrow y = z) \rightarrow (\exists T \forall y(y \in T \leftrightarrow \exists x(x \in S \wedge P(x,y)))))$

The first part of the implication simply establishes the predicate P as a function—if $P(x,y)$ and $P(x,z)$, then $y = z$.

So we can write **P5** less formally as

$\forall S \forall P((P \text{ is a function}) \rightarrow (\exists T \forall y(y \in T \leftrightarrow \exists x(x \in S \wedge P(x,y)))))$

Note that P is not required to be a **total** function—P is allowed to be undefined for some values of x.

The axiom of separation

Remember the difficulty that we encountered because of Frege's axiom of comprehension—the axiom was too liberal and allowed for the construction of sets with contradictory properties. Note, however, the similarity between the axiom of comprehension, and the following axiom, which follows from the axiom of replacement:

Axiom of separation $\forall S \forall Q \exists T \forall x (x \in T \leftrightarrow x \in S \wedge Q(x))$.

Notice that this axiom allows us to construct sets of elements with arbitrary properties, just as with Frege's axiom of comprehension. The difference in this case is that the elements of the new set **must be chosen from a preexisting set**.

In essence, given a set, the axiom of separation allows us to take arbitrary subsets of the set, provided there is a predicate expressing the selection criteria.

The axiom of separation follows from the axiom of replacement by taking $P(x, y)$ in the axiom of replacement to be $x = y \wedge Q(x)$. In this way, y is uniquely determined by x (hence P is a function), and P only holds for those x satisfying Q.

From the axiom of separation, we can easily show that the intersection of two sets always exists—for a given set T, we take $Q(x)$ to be the condition $x \in T$. The axiom then guarantees the existence of a subset of S containing those x for which $x \in T$—in other words, $S \cap T$.

Similarly, using the condition $x \notin T$, we can construct the difference of two sets.

The pairing axiom (**P3**) can also be derived from the axiom of replacement and the existence of the set $\{\emptyset, \{\emptyset\}\}$ (which can be established via the power set axiom).

Let U and V be sets. Then let $P(x, y)$ denote the following function on $\{\emptyset, \{\emptyset\}\}$:

$$P(x, y) = (x = \emptyset \wedge y = U) \vee (x = \{\emptyset\} \wedge y = V) \ .$$

Since the value of x in P uniquely determines the value of y, P denotes a function, and therefore, the set consisting of the y-values corresponding to each x-value in $\{\emptyset, \{\emptyset\}\}$ exists. But this set is precisely the set

$$\{U, V\} \ .$$

Thus, the pairing axiom is technically a derived axiom.

Intuitionism revisited—the axiom of choice

The axiom of choice will not play a major role in our future discussions, but it is worth taking the time to consider its meaning and implications, as this axiom divides the classicists from the intuitionists.

We first state without proof that it is possible to construct the cartesian product of two sets S and T:

$$S \times T := \{(s,t) \mid s \in S \wedge t \in T\} .$$

This notion is easily generalized to larger collections of sets:

$$S_1 \times \cdots \times S_n := \{(s_1, \ldots, s_n) \mid \forall i (s_i \in S_i)\} .$$

Indeed, we could consider countably infinite cross products:

$$\times_{i=1}^{\infty}(S_i) := \{(s_1, \ldots) \mid \forall i(s_i \in S_i)\} ,$$

or even uncountably infinite cross products (which are harder to represent explicitly).

Now, consider the following question: if S and T are non-empty, is $S \times T$ non-empty?

The answer is clearly yes. To exhibit an element of $S \times T$, it suffices simply to pick an element s of S and an element t of T (remember, both are non-empty). Then $(s, t) \in S \times T$.

What about longer cross products? If $S_1, \ldots S_n$ are all non-empty, must $S_1 \times \cdots \times S_n$ be non-empty as well?

Again, the answer is yes. To exhibit an element of $S_1 \times \cdots \times S_n$, we simply pick an element s_i from each S_i, since all of these sets are non-empty. Then $(s_1, \ldots, s_n) \in S_1 \times \cdots \times S_n$.

What about infinite cross products? Must they be non-empty, assuming the components of the cross product are all non-empty?

Well, to exhibit a member of the infinite cross product, it suffices to simply pick an element from each of the component sets, and then....

But that process could take a while....

If we can't produce an element of the cross product in finite time, can we be sure that one exists?

It turns out that, in this case, the existence of an element of the cross product (hence, that the cross product is non-empty) **cannot be proven** from the current axioms of ZF set theory.

The **Axiom of Choice** asserts the non-emptiness of cross products:

Axiom of Choice: Every non-empty collection of non-empty sets has a non-empty cross product.

The Axiom of Choice has been shown to be compatible with the other ZF axioms; in other words, adopting the Axiom of Choice will not create contradictions.

It turns out that denying the Axiom of Choice (i.e., adopting its negation) also will not create contradictions. So we are free to either accept or reject the axiom.

On what basis should we decide whether to accept the Axiom of Choice?

On the one hand, the Axiom of Choice allows us to prove some convenient theorems (e.g., every vector space has a basis).

On the other hand, the Axiom of Choice asserts that it is possible to make infinitely many choices all at once (choosing an element from a cross product amounts to choosing one element from each component).

But we could never (in the general case) write such a choice down in finite time (or in a finite-length proof)!

So we are asserting the existence of an object without being able to produce it. This is precisely the kind of reasoning that the intuitionist rejects; the Axiom of Choice is an existence axiom—it asserts the existence of an object without giving a procedure for producing it!

Thus an intuitionist must reject the Axiom of Choice; a classicist, on the other hand, would likely accept it.

The remaining axiom of ZF is the axiom of regularity, whose purpose, loosely speaking, is to prevent sets from containing themselves. More details 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 $\forall x(E(x) \rightarrow \neg P(x))$. 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: $\exists y(y + y = x)$. 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:

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

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 $E(u, v)$, where E 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 $\forall x(P(x) \vee S(x) \rightarrow F(x))$. (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 $q \to p$.

“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 $p \to q$.

"p is a necessary and sufficient condition for q" translates as $(p \to q) \wedge (q \to p)$, 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 $p \to q$.

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

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.