Basics of First-order Logic

Michael Weiss

May 12, 2019

These are adapted from a talk I gave at the Math for People meetup. They begin with an example chosen for amusement value, before getting down to business.

Later on I participated in a meetup devoted (initially) to going through Smullyan and Fitting's *Set Theory and the Continuum Problem* [12]. I expanded these notes to provide a background in mathematical logic. I also wrote "Notes on Smullyan & Fitting, and on Forcing". I decided against combining the two sets of notes; this set is independent of the second set.

1 The Dog Walking Ordinance

From "The Reader Over Your Shoulder" by Robert Graves and Alan Hodge, reproduced in *The World of Mathematics* (ed. James R. Newman), in Section XIII:3, written by Ernest Nagel.

From the Minutes of a Borough Council Meeting:

Councillor Trafford took exception to the proposed notice at the entrance of South Park: 'No dogs must be brought to this Park except on a lead.'

He pointed out that this order would not prevent an owner from releasing his pets, or pet, from a lead when once safely inside the Park.

The Chairman (Colonel Vine): What alternative wording would you propose, Councillor?

Councillor Trafford: 'Dogs are not allowed in this Park without leads.'

Councillor Hogg: Mr. Chairman, I object. The order should be addressed to the owners, not to the dogs.

Councillor Trafford: That is a nice point. Very well then: 'Owners of dogs are not allowed in this Park unless they keep them on leads.'

Councillor Hogg: Mr. Chairman, I object. Strictly speaking, this would prevent me as a dog-owner from leaving my dog in the back-garden at home and walking with Mrs. Hogg across the Park.

Councillor Trafford: Mr. Chairman, I suggest that our legalistic friend be asked to redraft the notice himself.

Councillor Hogg: Mr. Chairman, since Councillor Trafford finds it so difficult to improve on my original wording, I accept. 'Nobody without his dog on a lead is allowed in this Park.'

Councillor Trafford: Mr. Chairman, I object. Strictly speaking, this notice would prevent me, as a citizen, who owns no dog, from walking in the Park without first acquiring one.

Councillor Hogg (with some warmth): Very simply, then: 'Dogs must be led in this Park.'

Councillor Trafford: Mr. Chairman, I object: this reads as if it were a general injunction to the Borough to lead their dogs into the Park.

Councillor Hogg interposed a remark for which he was called to order; on

his withdrawing it, it was directed to be expunged from the Minutes.

The Chairman: Councillor Trafford, Councillor Hogg has had three tries; you have had only two...

Councillor Trafford: 'All dogs must be kept on leads in this Park.'

The Chairman: I see Councillor Hogg rising quite rightly to raise another objection. May I anticipate him with another amendment: 'All dogs in this Park must be kept on the lead.'

This draft was put to the vote and carried unanimously, with two abstentions.

2 The Ordinance in Logical Notation

Some notation:

Opd	:	person p owns dog d
Ldt	:	dog d is on a lead at time t
$Lpdt$	:	person p has dog d on a lead at time t
Adt	:	dog d arrives at the Park at time t
Pxt	:	x is in the Park at time t

All the following logical translations have an implicit enclosing "should": the ordinance says that a certain assertion *should* be true. (So-called modal logic has an explicit SHOULD quantifier, but we won't need it.)

No dogs must be brought to this Park except on a lead.

$$(\forall \text{ dog } d)(\forall \text{ time } t)(Adt \rightarrow Ldt)$$

Objection: this is consistent with $(\exists \text{ dog } d)(\exists \text{ time } t)[Pdt \wedge \neg Ldt]$.

Dogs are not allowed in this Park without leads.

$$(\forall \text{ dog } d)(\forall \text{ time } t)(Pdt \rightarrow Ldt)$$

Rejected for stylistic reasons, i.e., the predicate Ldt is addressed to the dogs. The ordinance should use $Lpdt$ instead.

Owners of dogs are not allowed in this Park unless they keep them on leads.

$$(\forall \text{ person } p)(\forall \text{ time } t)[Ppt \wedge [(\exists \text{ dog } d)Opd] \rightarrow Lpdt]$$

Objection: doesn't allow $(\exists \text{ time } t)(\exists \text{ person } p)(\exists \text{ dog } d)[Opd \wedge Ppt \wedge \neg Pdt \wedge \neg Lpdt]$.

Nobody without his dog on a lead is allowed in this Park.

$$(\forall \text{ person } p)(\forall \text{ time } t)[Ppt \rightarrow (\exists \text{ dog } d)[Opd \wedge Lpdt]]$$

Objection: doesn't allow

$$(\exists \text{ person } p)(\exists \text{ time } t)[Ppt \wedge \neg(\exists \text{ dog } d)Opd]$$

Dogs must be led in this Park.

Perhaps:

$$(\forall \text{ dog } d)(\forall \text{ time } t)[Pdt \wedge Ldt]$$

or maybe (if we want to avoid the use of Ldt)

$$(\forall \text{ dog } d)(\forall \text{ time } t)[(\exists \text{ person } p)Opd \rightarrow Pdt \wedge Lpdt]$$

both of which require all dogs to be in the Park at all times (except for ownerless dogs, with the second formulation). Or maybe:

$$(\forall \text{ dog } d)(\forall \text{ time } t)[Ldt \rightarrow Pdt]$$

i.e., dogs being walked on a lead must be in the Park.

All dogs must be kept on leads in this Park.

Same translations, same objection.

All dogs in this Park must be kept on the lead.

$$(\forall \text{ dog } d)(\forall \text{ time } t)[Pdt \rightarrow Ldt]$$

or maybe

$$(\forall \text{ dog } d)(\forall \text{ time } t)[Pdt \wedge (\exists \text{ person } p)Opd \rightarrow Lpdt]$$

This was deemed acceptable.

3 First-order Logic

First-order logic consists of syntax and semantics. The **syntax** specifies the "grammar" of a first-order language, so to speak: the rules that say when a string of symbols is a well-formed sentence, how to parse it, and when a list of sentences is a valid proof. A key point: syntax is something that can be checked *mechanically*: you could write a program to process it. Historically, the finitistic nature of syntax played an important role in philosophical debates about the nature of mathematics.

The **semantics** of first-order logic deals with questions of meaning. What does it mean to say that sentence is true or false? Let's say that GROUP are the basic axioms of group theory (associativity, etc.) When we say that a structure G is a group, we're making a statement about semantics: we're saying that all the axioms of GROUP hold true in G. In the jargon of logic, we say that G is a model of the axiom system GROUP.

3.1 Basic Notation

The basic notation consists of connectives, quantifiers, variables, relation and function symbols, constants, and punctuation. Punctuation is just three symbols: the parentheses and the comma.

Connectives

$\wedge$	:	and
$\vee$	:	or
$\neg$	:	not
$\rightarrow$	:	implies
$\leftrightarrow$	:	if and only if

Quantifiers

$\exists$	:	there exists
$\forall$	:	for all

Variables, relations, functions, constants

$x\ y\ z\ \ldots$	:	variables
$=$	:	equals
$P\ Q\ R\ <\ >\ \ldots$	:	relation symbols (aka predicates)
$f\ g\ h\ +\ \cdot\ \ldots$	:	function symbols
$0\ 1\ a\ b\ \ldots$	:	constants

Each first-order language has its own collection of relation and function symbols and constants, indeed is determined by this. (Except '=' is usually considered a basic logical symbol, common to all languages.) For example, so-called incidence geometry has unary predicate letters POINT and LINE, and a binary predicate letter ON. The other symbols (like $\wedge$, $\forall$, etc.) belong to the basic vocabulary. We assume there is an infinite supply of variables. If we're worried about someone with a finitist perspective, or we want to program this for a computer, we make do with something like x, x', x'', x''',...

3.2 Basic Syntax

The syntax of first-order logic specifies notions like **formula**, **free variable**, **bound variable**, **sentence**, etc. The definitions are generally recursive.

For example, a **term** is defined this way:

1. A constant is a term.
2. A variable is a term.
3. If f is an n-ary function symbol and $\tau_1, \ldots, \tau_n$ are terms, then $f(\tau_1, \ldots, \tau_n)$ is a term.

and a **formula** this way:

1. If P is an n-ary predicate and $\tau_1, \ldots, \tau_n$ are terms, then $P(\tau_1, \ldots, \tau_n)$ is a formula. More specifically, it's called an **atomic formula**.
2. If φ and ψ are formulas, the $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \to \psi$, and $\varphi \leftrightarrow \psi$ are formulas.
3. If φ is a formula and x is a variable, then $(\exists x)\varphi$ and $(\forall x)\varphi$ are formulas.

If you really want to get down into the weeds, you have to discuss all sorts of picky fine points. For example, '...' isn't present literally, since the ellipsis isn't part of the vocabulary. I already mentioned not really having an infinity of variable letters. We might write $\varphi(x)$ to indicate a formula with a free variable x; of course, the x might occur multiple times, as in the formula $x = x$. Etc.

The main point: all this can be done mechanically, and in principle could be checked by a computer program. (Not just "in principle": theorem-provers have to do this kind of thing for real.) For expositional purposes, it helps to have the "official" syntax, and the "vernacular". For example:

$x \neq y$	instead of	$\neg(x = y)$
$x < y$	instead of	$<(x, y)$
$x \geq y$	instead of	$<(y, x) \vee (x = y)$
$x + (y + z)$	instead of	$+(x, +(y, z))$
$(\exists x \in P)\varphi(x)$	instead of	$(\exists x)[P(x) \wedge \varphi(x)]$
$(\forall x \in P)\varphi(x)$	instead of	$(\forall x)[P(x) \to \varphi(x)]$

We could also make the "official" vocabulary lean and mean, regarding many symbols as abbreviations. Make $\exists$ official and treat $(\forall x)$ as an abbreviation for $\neg(\exists x)\neg$. (Or vice versa.) Do away with $\vee$ officially, introducing it on the black market in terms of $\neg$ and $\wedge$. This can make the formal development slicker.

Besides the notions of **terms** and **formulas**, we have **bound** and **free** variables, or better: bound and free occurrences of variables. I won't give a definition, just an illustration. Here is a formula where the occurrences of y are free and of x are bound.

$$(\forall x)(x < y) \vee (\exists x)(y < x)$$

A **closed formula** has no free variables. A closed formula is also called a **sentence**. A **closed term** has no variables at all, just constants and function symbols.

If $\varphi(x_1, \ldots, x_n)$ is a formula with free variables $x_1, \ldots, x_n$, and $\tau_1, \ldots, \tau_n$ are terms, then the result of substituting each τ_i for each x_i is denoted $\varphi(\tau_1, \ldots, \tau_n)$. People also write $\varphi^{x_1 \ldots x_n}_{\tau_1, \ldots \tau_n}$. It turns out that if you want to be picky, there are some pitfalls. Substituting x for y in $(\exists x)(x < y)$ would be wrong! But it's not hard, just tedious to come up with the right rules for legal substitution.

Formalizing a mathematical notion usually presents many choices. Here are two languages in which to talk about the elements of a group:

- We have a constant 1, a binary function symbol $\cdot$, and a unary function

symbol $^{-1}$. Sample formula (an axiom): $(\forall x)(x \cdot x^{-1} = 1)$.

- We have a binary function symbol $\cdot$. We let $\textsc{unit}(e)$ be an abbreviation for $(\forall x)x \cdot e = x$, and have axioms $(\exists e)\textsc{unit}(e)$ and $(\forall x)(\exists y)(\exists e)(x \cdot y = e \wedge \textsc{unit}(e))$.

We could also have a language with just a ternary relation symbol $\textsc{eq}(x, y, z)$, intended to express $x = y \cdot z$.

Second example: a graph. We could formalize this with just one binary predicate $\textsc{edge}(v, w)$. Or we could have two unary predicates $\textsc{vertex}(x)$ and $\textsc{edge}(e)$ and a binary predicate $\textsc{on}(v, e)$.

A **first-order theory** is, essentially, just a set of formulas; we call these the **axioms** of the theory. So, the theory GRAPH might include the axiom "$\textsc{on}(v, e) \rightarrow \textsc{vertex}(v) \wedge \textsc{edge}(e)$". Naturally, the formulation of the axioms depends on the language.

Already in the last example, you're probably thinking about what the symbols *mean*. If $\textsc{edge}(v, w)$ in the first language, shouldn't we have $(\exists e)[\textsc{on}(v, e) \wedge \textsc{on}(w, e)]$ in the second? Maybe you're thinking that the first language doesn't provide for multiple edges between a pair of vertices, so it's only good for talking about graphs without that.

But now we're straying from syntax into semantics.

3.3 Semantics

The main task of semantics: defining what it means for a sentence to be true, i.e., to be satisfied by a structure. A **structure** for a language contains enough "furniture" for the sentences to make sense—you can't ask if the axioms of geometry hold true until you have points and lines. A **model** of theory is one that satisfies all the theory's axioms.

Let $\mathcal{L}$ be a first order language. So $\mathcal{L}$ has some constants, some function symbols, and some predicates (aka relation symbols). A **structure S** for $\mathcal{L}$ consists of:

1. The universe of the structure, which is a class S.
2. An element of S for every constant c; I'll denote the element by $\overline{c}$.
3. An n-ary relation on S for every n-ary predicate P; I'll denote it by $\overline{P}$.
4. An n-ary function on S for every n-ary function symbol f; I'll denote it by $\overline{f}$.

In **S**, we can define an **interpretation** for any term $\tau(x_1, \ldots, x_n)$ and any formula $\varphi(x_1, \ldots, x_n)$ of $\mathcal{L}$. The interpretation of τ will be an n-ary function from S to S; let's denote it $\overline{\tau}$. The interpretation of φ will be an n-ary relation on S; we'll denote it $\overline{\varphi}$.

People don't generally use notation like $\overline{\tau}$ and $\overline{\varphi}$; instead they use the so-called satisfaction symbol $\models$. Let's say $\varphi(x_1, \ldots, x_n)$ is a formula with n free variables, and $(a_1, \ldots, a_n) \in S^n$. We write

$$\mathbf{S} \models \varphi(a_1, \ldots, a_n)$$

to mean that the relation $\overline{\varphi}$ holds for the n-tuple $(a_1, \ldots, a_n)$. (Sometimes we get sloppy and just write $S \models \varphi(a_1, \ldots, a_n)$ instead of $\mathbf{S} \models \varphi(a_1, \ldots, a_n)$. This is like when people say "the group G" instead of "the group $(G, \cdot)$.) We can also write

$$\mathbf{S} \models b = \tau(a_1, \ldots, a_n)$$

to mean that $\overline{\tau}$ applied to $(a_1, \ldots, a_n)$ yields b.

The definitions are recursive, of course. I probably don't need to spell out how terms are interpreted. As for formulas (and writing $\vec{a}$ for $a_1, \ldots, a_n$ and $\vec{b}$ for $b_1, \ldots, b_m$, to save space):

1. $\mathbf{S} \models \varphi(\vec{a}) \wedge \psi(\vec{b})$ iff $\mathbf{S} \models \varphi(\vec{a})$ and $\mathbf{S} \models \psi(\vec{b})$.
2. $\mathbf{S} \models \neg\varphi(\vec{a})$ iff it is not true that $\mathbf{S} \models \varphi(\vec{a})$.
3. $\mathbf{S} \models (\exists x)\varphi(x, \vec{a})$ iff for some $s \in S$, $\mathbf{S} \models \varphi(s, \vec{a})$.

In particular, if φ has no free variables, then $\overline{\varphi}$ is a 0-ary relation, which is just a truth-value: $\mathbf{S} \models \varphi$ is either true or false. (If you think of an n-ary relation as a function from S^n to the set {true,false}, this makes sense.)

Now let's step back and ask, what sort of expression is $\varphi(\vec{a})$? It's not a formula in the language $\mathcal{L}$ because the a's are not symbols in the language—they're elements of S. In some cases, you can get around this easily: in the theory of the natural numbers, we can write $1+1+\cdots+1$ (n 1's) to represent $n \in \mathbb{N}$. More generally, if we have a term representing every element of S, we're home free.

But to be truly general, one approach is to add a new constant to $\mathcal{L}$ for every element of S. Let's denote this augmented language by $\mathcal{L}_S$. Then things like $\varphi(\vec{a})$ are formulas in $\mathcal{L}_S$. We can even follow many authors by letting the constant for $a \in S$ be a itself. I will call these added constants **names**. So the constants of $\mathcal{L}_S$ consist of **built-in constants** (if any), plus all the names.

This does encroach on the line separating syntax from semantics. If, say, $S = \mathbb{R}$, then $\mathcal{L}_S$ is not machine checkable. Someone with a finitist or formalist philosophy would (probably) be happy with $\mathcal{L}$, provided it contains only finitely many symbols. $\mathcal{L}_{\mathbb{R}}$, not so much.

But if we treat the theory of first-order languages like any other theory—say like the theory of groups, or Hilbert spaces—then why not? A formula is a finite string of symbols. If the symbols are chosen from an infinite vocabulary, that's not a problem if we believe in infinite sets (or are willing to act like we do).

Even if you want to draw a sharp line separating $\mathcal{L}$ from $\mathcal{L}_S$, you still have to allow $\mathbf{S} \models \varphi(\vec{a})$ in the definition of satisfaction: the inductive nature of the $\exists$ clause demands it. Of course, $\models$ belongs to semantics.

Anyway, some terminology: A **pure formula** is a formula in $\mathcal{L}$, i.e., no names appear in it. A **sentence** is a closed formula. I prefer the terms **statement** or **assertion**, but **sentence** seems to be standard. I will use *assertion* specifically for closed formulas of $\mathcal{L}_S$, i.e., to highlight that names are permitted. Note that pure formulas can still have constants, just so long as they belong to the base language $\mathcal{L}$—what I call "built-in constants". For example, when we describe the first-order theory of real numbers in §4.4, we'll include constants 0 and 1. So $0 < 1$ is a pure sentence, but $e < \pi$ is an assertion.

Things get a lot hairier if we want to allow infinitely long formulas. Logicians have investigated stuff like that, but that's outside the realm of first-order logic.

Finally, an extension of the $\models$ notation to make things smoother. If $\varphi(\vec{x})$ has free variables $\vec{x}$, then $\varphi(\vec{x})$ doesn't usually have a definite truth value: I don't know if $x = y$ is true or false until you tell me what x and y are. But I know that $x = x$ regardless of which x you choose. Let's agree that

$$\mathbf{S} \models \varphi(x_1, \ldots, x_n)$$

means the same as

$$\mathbf{S} \models (\forall x_1) \ldots (\forall x_n) \varphi(x_1, \ldots, x_n)$$

(for short: $\mathbf{S} \models (\forall \vec{x}) \varphi(\vec{x})$).

Another extension: say $\mathcal{T}$ is a set of formulas. We write

$$\mathbf{S} \models \mathcal{T}$$

to mean the same as

$$\mathbf{S} \models \varphi \text{ for every } \varphi \in \mathcal{T}$$

and we say $\mathbf{S}$ is a **model** of $\mathcal{T}$.

3.4 Rules of Inference; Theories; Proofs

Now we return to syntax, but syntax inspired by semantics. Suppose, for example, that

$$\mathbf{S} \models \varphi \to \psi$$

and

$$\mathbf{S} \models \varphi$$

(I've let the $(\vec{a})$ and $(\vec{b})$ be implicit, as people usually do when they can get away with it.) Then we *must* have

$$\mathbf{S} \models \psi$$

Likewise, if τ is a term and

$$\mathbf{S} \models (\forall x)\varphi(x)$$

then we *must* have

$$\mathbf{S} \models \varphi(\tau)$$

This suggests two **rules of inference**:

Modus ponens. For any formulas φ and ψ, this says:

$$\begin{array}{lcl} \text{From} & : & \varphi \to \psi \\ \text{and} & : & \varphi \\ \text{infer} & : & \psi \end{array}$$

Particularization. For a formula $\varphi(x)$ and a term τ, this says:

$$\begin{array}{lcl} \text{From} & : & (\forall x)\varphi(x) \\ \text{infer} & : & \varphi(\tau) \end{array}$$

There are also formulas that must be satisfied in *any* structure. Number one example:

$$\varphi \vee \neg\varphi$$

A close second is $(\forall x)(x = x)$. These are called **logical axioms**. There are a bunch of other logical axioms I won't bother to list. (The list of logical axioms and of inference rules isn't standardized; you can achieve the same end results in many different ways.)

Key point: although these rules are inspired by semantics, they are purely syntactical: a machine could check if a formula fits the $\varphi \vee \neg\varphi$ template, or if an ordered triple of formulas matches the modus ponens template. In general, matching an inference rule or being a logical axiom is machine checkable.

Definition: a **theory** $\mathcal{T}$ for a language $\mathcal{L}$ is a set of formulas of $\mathcal{L}$. (We also say that $\mathcal{L}$ is the **language of** $\mathcal{T}$, and write $\mathcal{L}(\mathcal{T})$ for the language of a theory $\mathcal{T}$.) The formulas in $\mathcal{T}$ are called **non-logical axioms**. (I also like the term **postulate**, following Euclid, but few authors use it.)

Definition: a **proof** in a theory $\mathcal{T}$ is a finite list of formulas such that every formula in the list is

1. either an axiom (logical or non-logical), or
2. follows from earlier formulas on the list by a rule of inference

and a **theorem** is the last line of a proof.

Notation: if $\mathcal{T}$ is a theory, we write

$$T \vdash \varphi$$

to mean φ is a theorem of $\mathcal{T}$ (pronounced "$\mathcal{T}$ yields φ").

To repeat a definition from the last section: if $\mathbf{S} \models \varphi$ for every φ in $\mathcal{T}$, we say $\mathbf{S}$ is a **model** of $\mathcal{T}$, and write $\mathbf{S} \models \mathcal{T}$.

Now, the whole point of the logical axioms and rules of inference is to make valid deductions. So the following claim *better* be true:

> If $\mathbf{S} \models \mathcal{T}$ and $\mathcal{T} \vdash \varphi$, then $\mathbf{S} \models \varphi$.

In other words:

> If we can prove (using the rules of logic) that φ follows from the axioms of $\mathcal{T}$, and a structure $\mathbf{S}$ is a model of $\mathcal{T}$, then $\mathbf{S}$ also satisfies φ.

If this didn't hold, we'd have a serious flaw in the rules of logic!

Picky points: some subtleties surround the proper treatment of quantifiers. Roughly speaking, we want a formal equivalent to arguments like this: "We know that $\varphi(x)$ holds for all x, so let x be arbitrary; then $\varphi(x)$ holds." Or like this: "We know that there exists an x for which $\varphi(x)$ is true, so let c be such an x." Different authors handle this in different ways. Here's a hint at some of the issues. We'd like to apply modus ponens to this pair of formulas:

$$(\forall x)(\varphi(x) \to \psi(x))$$
$$(\forall x)\varphi(x)$$

But this doesn't fit the "$P \to Q$, P" template. Of course for any term τ, we can deduce $\varphi(\tau) \to \psi(\tau)$ and $\varphi(\tau)$, and thus get $\psi(\tau)$ by modus ponens. But that isn't quite $(\forall x)\psi(x)$. The "implicit universal quantification" convention for $\mathbf{S} \models \psi(x)$ suggests a way out: we add a new rule of inference to infer $(\forall x)\psi(x)$ from $\psi(x)$. When hedged with the right caveats this works well. It adheres to the usual mathematical practice of saying things like "In an abelian group, $xy = yx$" instead of "In an abelian group, for all x and y, $xy = yx$".

3.5 Consistency, Completeness, Decidability

Fact: *no* structure can satisfy $\varphi \wedge \neg\varphi$, for any closed formula φ. We *never* have $\mathbf{S} \models \varphi \wedge \neg\varphi$.

So if $\mathcal{T} \vdash \varphi \wedge \neg\varphi$, then $\mathcal{T}$ can never have a model. (If we had $\mathbf{S} \models \mathcal{T}$, then since $\mathcal{T} \vdash \varphi \wedge \neg\varphi$, we'd also have $\mathbf{S} \models \varphi \wedge \neg\varphi$, which we just saw is impossible.)

We say $\mathcal{T}$ is **inconsistent** if $\mathcal{T} \vdash \varphi \wedge \neg\varphi$ for some closed formula φ. Otherwise, $\mathcal{T}$ is **consistent**.

So an inconsistent theory can never have a model. Does a consistent theory always have a model? Answer: yes. This is the famous Gödel completeness theorem. Not easy to prove.

The term **completeness** is also used in a very different way. $\mathcal{T}$ is **complete** if for any closed formula φ in the language of $\mathcal{T}$, either $\mathcal{T} \vdash \varphi$ or $\mathcal{T} \vdash \neg\varphi$. In other words, $\mathcal{T}$ answers every question about models of $\mathcal{T}$ one way or the other.

Complete theories are kind of rare. Most theories are "deliberately incomplete". For example, the theory of groups is incomplete, since it doesn't answer the question "Does $(\forall x)(\forall y)(x \cdot y = y \cdot x)$ hold?". That's because we don't *want* it to answer that question: we want the theory to include both abelian and non-abelian groups.

On the other hand, we'd love to have a complete consistent first-order theory for number theory. Now, technically you can get one, at least if you believe the phrase "the set of all closed formulas φ that are true of $\mathbb{N}$" means something. I.e., the set $\mathcal{T} = \{\varphi : \mathbb{N} \models \varphi\}$ is a complete consistent theory. (Really I should write $(\mathbb{N}, 0, 1, +, \cdot)$ instead of $\mathbb{N}$, but people usually don't. Abuse of notation.) $\mathcal{T}$ is so-called "true arithmetic".

Problem is, we have no way of telling, in general, whether something is an

axiom of true arithmetic. So it's useless for proving theorems of number theory.

If $\mathcal{P}$ is a set of axioms for the language of number theory that is **recursive**—there's a program to check whether any φ belongs to $\mathcal{P}$—then $\mathcal{P}$ is either inconsistent or incomplete. That's Gödel's famous incompleteness theorem. (Picky point: $\mathcal{P}$ must be "sufficiently rich" for incompleteness to apply. It turns out that if you eliminate multiplication from the language, then the resulting impoverished theory is complete. But then, there's very little you can say with such a limited vocabulary!)

Finally, a theory $\mathcal{T}$ is **decidable** if there's a program that will tell you, for any closed φ, if $\mathcal{T} \vdash \varphi$ or not. To elaborate: suppose $\mathcal{T}$ is recursive, so you have a mechanical way to tell if a formula is an axiom of $\mathcal{T}$ or not. Then you have a mechanical way to tell if a list of formulas is a *proof* or not. But you don't necessarily have a way to tell if a formula is a *theorem* or not. Just because you haven't found a proof yet, doesn't mean there isn't one out there!

4 Examples

It's a pretty standard convention to let outermost universal quantifiers be implicit. That is, instead of writing $(\forall x)(\forall y)[x < y \vee x = y \vee y < x]$, we just write $x < y \vee x = y \vee y < x$. Of course, with something like $(\exists e)(\forall x)[e \cdot x = x]$, the inner $(\forall x)$ cannot be omitted. If we simply wrote, $(\exists e)[e{\cdot}x = x]$, that means (by the convention) the same as $(\forall x)(\exists e)[e{\cdot}x = x]$, which is a weaker assertion: it permits the choice of e to depend on the choice of x.

4.1 First-order Theory of a Graph

Simplest approach: let the edges be implicit, so the variables range *only* over vertices. Just one predicate $x \sim y$: x and y are connected by an edge. Postulates:

$$\begin{array}{lcl} \text{Symmetry} & : & x \sim y \to y \sim x \\ \text{Irreflexive} & : & \neg x \sim x \end{array}$$

A model of this theory is a graph, with undirected edges, at most one edge between any two vertices, and no loops (edges from a vertex to itself).

As an example of a theorem, we have non-transitivity, if there are any edges at all:

$$(\exists x, y)(x \sim y) \to \neg(\forall x, y, z)(x \sim y \sim z \to x \sim z)$$

(heavy use of vernacular). Proof: let a, b be the two nodes with $a \sim b$. By symmetry, $a \sim b \sim a$, but $a \not\sim a$ by irreflexivity.

Turning this into a formal proof poses a challenge—the quantifier issue rears its head. Consider the transition from $(\exists x, y)\varphi(x, y)$ to "let a, b satisfy $\varphi(a, b)$". Some authors formalize this sort of thing as a rule of inference; dotting all the i's takes some work. Other authors treat $\exists x$ as an abbreviation for $\neg(\forall x)\neg$, and have logical axiom schemas like this

$$(\forall x)[\varphi(x) \to \psi(x)] \to [(\forall x)\varphi(x) \to (\forall x)\psi(x)]$$

plus an inference rule that says "infer $(\forall x)\varphi(x)$ from $\varphi(x)$". I won't go into details.

Many things you'd like to say about graphs can't even be formulated in this language. We have no means to talk about natural numbers; thus, "x is connected to y via a chain of edges" is beyond our ken. We can say, "x is connected to y via a chain of 100 edges" with a formula containing 98 existential quantifiers, but not the general statement "$(\exists n \in \mathbb{N})\ldots$".

We can't say, "graphs G and H are isomorphic", or even talk about more than one graph at a time. In short, almost everything a graph-theorist wants to say demands a more expressive language (typically set theory).

Sometimes there are work-arounds. Suppose we want to say, "G can be colored with four colors." We can't in our language. But let's add four predicate letters to the language: RED, BLUE, GREEN, YELLOW. Then add some more postulates:

Every vertex has a color	:	$\text{RED}\, x \vee \text{BLUE}\, x \vee \text{GREEN}\, x \vee \text{YELLOW}\, x$
And only one color	:	$(\text{RED}\, x \to \neg(\text{BLUE}\, x \vee \text{GREEN}\, x \vee \text{YELLOW}\, x))$ $\wedge(\text{BLUE}\, x \to \neg(\text{GREEN}\, x \vee \text{YELLOW}\, x))$ $\wedge(\text{GREEN}\, x \to \neg\text{YELLOW}\, x)$
Adjacent vertices have different colors	:	$x \sim y \to \neg(\text{RED}\, x \wedge \text{RED}\, y)$ $\wedge\neg(\text{BLUE}\, x \wedge \text{BLUE}\, y)$ $\wedge\neg(\text{GREEN}\, x \wedge \text{GREEN}\, y)$ $\wedge\neg(\text{YELLOW}\, x \wedge \text{YELLOW}\, y)$

So a model of this augmented theory is a graph that has been colored with four colors.

4.2 Linear Orderings

The language of linear orderings, $\mathcal{L}(\text{LO})$, has a single predicate $x < y$ and the following postulates:

$$\neg(x < x)$$
$$(x < y < z) \to (x < z)$$
$$x < y \vee x = y \vee y < x$$

An easy consequence of the first two postulates: $(x < y) \to \neg(y < x)$.

We use $x \leq y$ as an abbreviation for $x < y \vee x = y$. As a mild variant, we can make $\leq$ fundamental, in which case the postulates are:

$$x \leq x$$
$$(x \leq y \wedge y \leq x) \rightarrow (x = y)$$
$$(x \leq y \leq z) \rightarrow (x \leq z)$$
$$x \leq y \vee y \leq x$$

Omitting the last postulate (in either formulation) gives us the theory of partial orderings.

A linear ordering is *dense* if it satisfies the further postulate

$$(\forall x < z)(\exists y)[x < y < z]$$

Maximum and minimum elements are defined in the usual way; these are both *endpoints*. The theory of dense linear orderings without endpoints (denoted DLO) enjoys several pleasant properties. First, any two countable dense linear orderings without endpoints are order-isomorphic. DLO is complete and decidable. It also admits *elimination of quantifiers*, which means that given any $\varphi(\vec{x})$ in $\mathcal{L}(\mathrm{LO})$, there is a quantifier-free $\psi(\vec{x})$ in $\mathcal{L}(\mathrm{LO})$ such that

$$\mathrm{DLO} \vdash \varphi(\vec{x}) \leftrightarrow \psi(\vec{x})$$

As one of the simplest examples, the formula $(\exists y)[x < y < z]$ is equivalent in DLO to the quantifier-free formula $x < z$. Perhaps even simpler: $(\exists y)x < y$ is equivalent in DLO to $x = x$. (Remember that DLO includes "without endpoints".)

Cantor proved the order isomorphism of all countable dense linear orderings without endpoints; nowadays one demonstrates this with the so-called back-and-forth method. The idea is simple. Assume we have enumerations of the elements of the DLOs A and B (not in increasing order, of course). We construct a new pair of enumerations, $a_1, a_2, \ldots$ and $b_1, b_2, \ldots$, such that if we pair up a_n with b_n for all n, this pairing with be an order isomorphism.

Step 1: pick any element of A for a_1, and any element of B for b_1. Step $n+1$: we assume $a_1, \ldots a_n$ and $b_1, \ldots b_n$ have already been chosen and have the same ordering, i.e., if we arranged both the a_i's and the b_i's in increasing order, they would remain paired up the same way. At odd-numbered steps, pick the first unpaired element in the enumeration of A to be a_{n+1}. Because of density and the lack of endpoints, B must have an unpaired element "in the right place". Specifically, if a_{n+1} is greater than all the a_i $(i = 1, \ldots, n)$, then we want an element of B that is greater than all the b_i $(i = 1, \ldots, n)$. Likewise for a_{n+1} that is less than all of the a_i. Finally, if a_{n+1} is between a_i and a_j for some i and j (both in range from 1 to n), then we want an element of B that is between b_i and b_j (same i and j). Pick b_{n+1} to be the first element "in the right place" in the enumeration of B. At even-numbered steps we do the same thing, except switching the roles of A and B.

The existence of lots of order-isomorphisms lies at the heart of the pleasant DLO properties cited above. If we look at $\mathbb{Q}$, it's easy enough to construct a piecewise linear order-isomorphism $\mathbb{Q} \to \mathbb{Q}$ mapping any strictly increasing sequence of n rational numbers to any other strictly increasing sequence of n rational numbers[1]. So the same is true for any countable model of DLO. Alternately, you can adapt the back-and-forth proof to show this. As a nearly trivial consequence, if $\varphi(x)$ is a formula in $\mathcal{L}(\mathrm{LO})$ with one free variable, and if a countable DLO model $Q \models \varphi(a)$ for $a \in Q$, then $Q \models \varphi(b)$ for *all* $b \in Q$. More generally, if $\vec{a}$ is a strictly increasing sequence of n elements of Q, and $Q \models \varphi(\vec{a})$ (with $\varphi(\vec{x})$ now a formula with n free variables), then $Q \models \varphi(\vec{b})$ for any other strictly increasing sequence $\vec{b}$ of length n. Loosely speaking, in $\mathcal{L}(\mathrm{LO})$ we can express only the following properties of a finite subset of a countable model of DLO: its cardinality and the order of its elements. This should make the elimination quantifiers at least plausible.

[1]Note that the arithmetic operations do not belong to $\mathcal{L}(\mathrm{LO})$, so the construction of this function occurs "outside DLO". But we can still use the existence of such functions to reason about the theory.

Turning from DLO just to LO, we have a much more varied landscape. As with GRAPH, much of what you'd like to say can't be expressed in $\mathcal{L}(\text{LO})$. You *can* say formally that y is an immediate successor (or predecessor) of x, and prove with LO that immediate successors and predecessors are unique (when they exist). But you can't say, using just $\mathcal{L}(\text{LO})$, that only a finite number of elements lie between x and y. You can't characterize the order type of $\mathbb{N}$ with postulates, nor that of $\mathbb{Z}$. The notion of well-ordering is likewise beyond the grasp of $\mathcal{L}(\text{LO})$.

4.3 Peano Arithmetic

In 1889, Giuseppe Peano gave a set of first-order postulates for the arithmetic of natural numbers. Here's a modern version.

Language:

0 and 1	:	constants
$+$ and $\cdot$	:	addition and multiplication function symbols

The universe is the set of natural numbers, so variables range (implicitly) over them.

Postulates:

0 is first	:	$\neg(\exists x)x + 1 = 0$
No number its own successor	:	$\neg(\exists x)x + 1 = x$
Identity property of 0	:	$x + 0 = x \wedge 0 + x = x$
0 and multiplication	:	$x \cdot 0 = 0 \wedge 0 \cdot x = 0$
Identity property of 1	:	$x \cdot 1 = x \wedge 1 \cdot x = x$
Inductive properties of addition	:	$(x + 1) + y = (x + y) + 1$ $x + (y + 1) = (x + y) + 1$
Inductive properties of multiplication	:	$(x + 1) \cdot y = x \cdot y + y$ $x \cdot (y + 1) = x \cdot y + x$
Induction postulates	:	for any formula $\varphi(x)$, we have $[\varphi(0) \wedge (\forall x)(\varphi(x) \to \varphi(x + 1))]$ $\to (\forall x)\varphi(x)$

The induction postulates form an infinite family of postulates. For example, this $\varphi(x)$ is used in an inductive proof of commutativity of addition:

$$(\forall y)[x + y = y + x]$$

and this $\varphi(x)$ is used to prove that every natural number is even or odd:

$$(\exists y)[y + y = x \vee (y + y) + 1 = x]$$

This system is always called Peano arithmetic and traditionally denoted PA.

The induction postulates of PA deal only with properties that can be expressed in the language of PA. Loosely speaking, the family of all *informal* induction postulates examines every subset N of $\mathbb{N}$, and discovers that except for $\mathbb{N}$ itself, either the property $0 \in N$ fails, or else the property $(\forall x)[x \in N \to (x + 1) \in N]$ fails. But Peano arithmetic conducts this examination only over the smaller countable collection of subsets definable by formulas $\varphi(x)$. That explains why there are models of PA that are not isomorphic to $\mathbb{N}$ (so-called non-standard models).

PA is surprisingly expressive—in principle, you could rewrite nearly all of typical undergraduate number theory in its language. (It would be unreadable, but so what.) Much that doesn't look (at first glance) expressible, *is*. For example, the theory of quadratic number fields deals with things like $a + b\sqrt{d}$, with $a, b \in \mathbb{Q}$. Rational numbers can be replaced with pairs of integers; we can manipulate equations to get rid of square roots, and to replace integers with natural numbers. We have the flexibility to replace a single equation with multiple equations and inequalities (using the boolean connectives), which helps.

Or take the prime number theorem. This is a statement about limits; using power series, rational approximations, and stuff like that, we can shoehorn it into PA. We can even handle many contour integrals in the complex plane (use Riemann sums).

Gödel came up with a trick for dealing with an arbitrary finite sequence $(i_1, \ldots, i_n)$ of elements of $\mathbb{N}$: code it as $p_1^{i_1} \cdots p_n^{i_n}$, where p_i is the i-th prime. (The details are a bit complicated; see §8.) Unlike what we encountered with GRAPH, we can say things like "for all finite sequences, such-and-such holds"— we can do this with a single formula, instead of resorting to an infinite sequence of them. Or "there exists a finite sequence such that...", which is inexpressible in GRAPH in any manner.

Of course, being able to *express* something in PA doesn't mean we can *prove* it in PA.

4.4 Real Numbers

The first-order theory of real numbers has constants 0 and 1, binary function symbols + and $\cdot$, and the binary predicate $<$. In addition to the usual field

axioms, we have the order axioms:

$$\begin{aligned}
&x < y \vee x = y \vee y < x \\
&x < y \wedge y < z \to x < z \\
&\neg(x < x) \\
&x < y \to x + z < y + z \\
&x < y \wedge 0 < z \to x \cdot z < y \cdot z \\
&0 < x \to (\exists y) y \cdot y = x
\end{aligned}$$

plus for every odd degree, an axiom that says that polynomials of odd degree have roots. I.e., for $n = 1, 3, \ldots$ an axiom of the form

$$a_n \neq 0 \to (\exists x) a_n \cdot x^n + \cdots + a_0 = 0$$

Here the a_i's are variables of the language, and for each n we explicitly write out the polynomial. For example, for $n = 3$:

$$a \neq 0 \to (\exists x)[(((a \cdot (x \cdot (x \cdot x))) + (b \cdot (x \cdot x))) + (c \cdot x)) + d = 0]$$

with the implicit universal quantification $(\forall a)(\forall b)(\forall c)(\forall d)$ out in front.

Technically, these are the axioms for a **real-closed field**. If you're wondering what happened to least upper bounds, or Dedekind cuts, those notions require a modicum of set theory. The real-closed field axioms capture the "algebraic part" of the theory of real numbers.

Surprisingly, this set of axioms is complete, consistent, and decidable! This result is due to Tarski. So any "purely algebraic" question about the real numbers can be decided mechanically. Using analytic geometry, this includes all of high-school geometry. We can say some fairly interesting stuff in this language.

However, almost all interesting statements of elementary analysis cannot be expressed in this language. There's no way to talk about an arbitrary continuous function, for example. We could augment the language, say by

adding predicates NUMBER and FUNCTION and a binary function symbol EVAL: EVAL(f, x) gives $f(x)$ when f is a function and x is a number. First-order syntax does not permit $x(y)$ for variables x and y, so we have to do things this way or leave first-order logic.

In this augmented language, we can state the definition of continuity, the intermediate value theorem, and the maximum value theorem. But compactness is a problem: how do we define an open cover? We could replace open sets with their characteristic functions, but we have no way to discuss arbitrary sets of functions.

In short, pushing in this direction starts to look more and more like set theory.

4.5 Groups

We already introduced an axiom system for the first-order theory GROUP: a constant 1, a binary function symbol $\cdot$, a unary function symbol $^{-1}$, and axioms

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$
$$x \cdot 1 = x$$
$$x \cdot x^{-1} = 1$$

So if $\mathbf{G} = (G, \circ) \models$ GROUP, then G is a group under the operation $\circ$.

A minor variation: get rid of $^{-1}$, replacing it with the axioms

$$(\forall x)(\exists y) x \cdot y = 1$$
$$x \cdot y = 1 \wedge x \cdot z = 1 \rightarrow y = z$$

It turns out that the second axiom is redundant, but this is an "algebraic" fact rather than a "logical" one. The vernacular expression

$$(\forall x)(\exists! y) x \cdot y = 1$$

encompasses both formulas, where $\exists!$ is read "there exists uniquely". A mechanical rule can be give for expanding any formula with $\exists!$ into formulas without it.

A rather different axiomatization for GROUP: we have a single ternary relation EQ. Motivation: a group is completely determined by its multiplication table. $\mathbf{G} \models \text{EQ}(x, y, z)$ when $x = y \circ z$. The associative axiom becomes a bit of a mess:

$$\text{EQ}(u, x, y) \wedge \text{EQ}(v, y, z) \wedge \text{EQ}(s, u, z) \wedge \text{EQ}(t, x, v) \rightarrow s = t$$

I won't bother with the remaining axioms.

Now, the axioms GROUP are the first-order axioms of *a* group. Here's an example of theorem we can prove:

$$(\forall x)xx = 1 \rightarrow (\forall x, y)xy = yx$$

Proof:

$$\begin{aligned} (xy)(xy) &= 1 \rightarrow \\ x(xyxy)y &= xy \rightarrow \\ (xx)yx(yy) &= xy \rightarrow \\ yx &= xy \end{aligned}$$

A formal proof would be longer; you'd have to be much more explicit about uses of associativity. Quantification isn't much of a problem, though.

For a group theorist, this language is painfully impoverished. We lack natural numbers: we can't say, "the group is finite" or "g has finite order", just things like "the group has less than 100 elements" or "g has order less than 100". We have no way to talk about subgroups or homomorphisms. As with the theory of the real numbers, to get to interesting material we'd need to add a bunch of stuff from set theory.

4.6 Set Theory

As we've seen repeatedly, to get to the "good stuff" in math, you almost always need some set theory. Zermelo-Fraenkel set theory (ZF), plus the axiom of choice (AC; ZF+AC=ZFC) has become the standard first-order axiom system for set theory.

ZF boasts a spartan vocabulary: just $\in$ plus the basic symbols of first-order logic. I won't write out all the axioms, just a few to give the flavor. Historically, ZFC arose (partly[2]) in response to the famous paradoxes. Zermelo and others diagnosed the predicament this way: we must drop the assumption that every property defines a set. That way, Russell's paradox lies: $R = \{x : x \notin x\}$ may *look* like it defines a set, but it doesn't.

Without this free-wheeling creation of sets, we need axioms to tell us which sets exist. All except two of the axioms of ZFC fall into this category.

The two exceptions are Extensionality and Foundation. Extensionality says:

$$(\forall z)(z \in x \leftrightarrow z \in y) \rightarrow x = y$$

Using the defined symbol $\subseteq$ (defined as $(\forall z)(z \in x \rightarrow z \in y)$), we could also write this as "$x \subseteq y \wedge y \subseteq x \rightarrow x = y$".

Foundation says, loosely, that everything is built up from the empty set. This isn't absurd, because we can use various techniques to "emulate" things like natural numbers, ordered pairs, functions, etc. Dedekind cuts taught mathematicians to think of real numbers as sets of rational numbers. Landau's book *Foundations of Analysis* built up the real numbers starting from the natural numbers. John von Neumann came up with a clever definition for the natural numbers: $0 = \varnothing$, $1 = \{0\}$, $2 = \{0, 1\}$, $3 = \{0, 1, 2\}$,... (In ZF, the set $\{0, 1, 2, \ldots\}$ is called the set of finite ordinals; following Cantor, it's denoted ω.) Kuratowki provided this trick

[2]Moore [6, 7] argues that Zermelo's primary motivation was to shore up his proof of the well-ordering theorem.

for ordered pairs: $\langle x, y\rangle = \{\{x\}, \{x, y\}\}$. If ordered pairs are here, can functions be far behind? Etc.

Here's the formal statement of Foundation:

$$(\forall x)(\exists y) y \cap x = \varnothing$$

Of course, $\cap$ and $\varnothing$ are defined terms. Foundation makes the formal development more elegant, but some authors just regard it as a "nice to have" and don't include it in ZFC.

The other axioms are EmptySet, Pairing, Union, Infinity, PowerSet, Replacement, and Choice. I won't bother with most of these; I'll just say something about the last three.

PowerSet says:

$$(\forall x)(\exists p)(\forall s)[s \in p \leftrightarrow s \subseteq x]$$

Or in other words, $p = \{s : s \subseteq x\}$, just the kind of set-builder definition that was uncritically accepted before the paradoxes. The power set of x is denoted $P(x)$.

Replacement (sometimes called Substitution) isn't a single axiom, but an **axiom schema**. We've seen this sort of thing before: the induction axioms of PA are an axiom schema. Suppose we have a formula $\varphi(x, y, \vec{t})$. Here's the idea of Replacement. Suppose for some particular choice $\vec{t} = \vec{a}$, the formula $\varphi(x, y, \vec{t})$ defines y as a partial function $\hat{\varphi}$ of x: given x, there is at most one y making $\varphi(x, y, \vec{a})$ true. Then the range of $\hat{\varphi}$ on any set is a set. It's not a bad exercise to try to write all this down in the vernacular. (Going all the way to the formal language is for masochists.)

Intuition for Replacement: if u is a set, then the range of $\hat{\varphi}$ on u is "no bigger" than u, and so ought to be a set also. That's a bit deceptive: once we have the range, we can apply Union to it to get much bigger sets! For example, it's possible in ZF to define the sequence $\omega, P(\omega), P(P(\omega)), \ldots$ We can express this sequence as the range of a partial function defined by a

formula φ. Taking the union, we have a huge set—its cardinality is at least $\aleph_\omega$.

Replacement has a consequence called Separation: if $\varphi(x, \vec{t})$ is a formula with free variables as shown, then for any set u and any sets $\vec{a}$, the following is also a set: $\{z \in u : \varphi(z, \vec{a})\}$. (Like Replacement, Separation is an schema.) According to Separation, if you have a property defined by a formula $\varphi(z, \vec{a})$, then you can "separate out" the elements of given set u satisfying this property, and get a new set. Zermelo originally include Separation but not Replacement in his axioms, but when Fraenkel pointed out the need for Replacement, Zermelo agreed.

Choice (AC) was the scene for major philosophical combat early in the 20th century. It's easier to express than Replacement:

$$(\forall x)(\exists \text{ function } c)[s \subseteq x \wedge s \neq \varnothing \to c(s) \in s]$$

There's a bit of vernacular ("function", "$c(s)$") but it's easily replaced. Unlike every other "set existence" axiom of ZFC, we can't define c with set-builder notation, or indeed give any other explicit description of the choice function. So people didn't like AC. In 1938, Gödel showed if ZF is consistent, then ZFC is too. (For good measure, he also showed that ZFC+GCH is consistent (assuming ZF is), where GCH is the generalized continuum hypothesis: $2^{\aleph_\alpha} = \aleph_{\alpha+1}$ for all α.) That calmed things down somewhat. Moore's book [7] treats the history of the Axiom of Choice.

Gödel's relative consistency results used the method of inner models; I'll discuss this in §6. For now, I'll just mention the central idea: the notion of a *constructible set*. The so-called class of all constructible sets is denoted L; the class of *all* sets is denoted V. In ZF, the word *class* has no formal meaning, but it provides a snappy way to say some things. For example "$V = L$" says that all sets are constructible. I'll say more about L in §10.3.

The word *class* proved so convenient (even before Gödel's 1938 paper) that a formal theory, call NBG, was invented to give a home for it. (NBG =

von Neumann, Bernays, Gödel.) In NBG, we still have only the symbol $\in$ plus the basic logical symbols. However, certain members of the "universe" have the left-hand side of $\in$ barred to them. If $x \in y$, then we say x is a **set**; anything that's not a set is a **proper class**. So proper classes can have sets as elements, but cannot themselves be elements. The term **class** encompasses both sets and proper classes; in NBG, the variables range over classes.

Here's how NBG skirts around Russell's paradox. We can still write the formal expression $R = \{x : x \notin x\}$. This defines the class R, which is the class of all sets that are not elements of themselves. Is $R \in R$? No, because if it were, it would have to be a set that was not an element of itself. OK, if $R \notin R$, doesn't that mean that R satisfies the condition to be an element of R? No, not if R is a proper class—R contains only sets, no proper classes allowed!

As it happens, Foundation (retained in NBG) implies that $x \notin x$ for all sets x, so R just turns out to be the class of all sets, V. (Statements like $R = V$ and $V = L$ have formal meanings in NBG.)

It turns out that NBG is a so-called conservative extension of ZFC: any formula of NBG that "talks only about sets" is provable in NBG iff it is provable in ZF. We'll see exactly what this means in §5.

5 Relativization and "Emulation"

Let's suppose that INTEGERS is a first-order theory for the integers. (Maybe we add a new binary predicate $<$ and make some modifications to PA.) Variables range over all the integers, so if we interpret $(\exists x)\varphi(x)$ in $\mathbb{Z}$, we're saying that there is an integer x satisfying $\varphi(x)$.

Now, $\mathbb{N}$ sits inside $\mathbb{Z}$, so we should be able to talk about $\mathbb{N}$ "inside INTE-

GERS". Here's how we do that: if φ is a formula in $\mathcal{L}$(INTEGERS), then we let $\varphi^{\mathbb{N}}$ be what you get by replacing all $(\forall x)$ with $(\forall x \geq 0)$, and all $(\exists x)$ with $(\exists x \geq 0)$. Of course, these are vernacular expressions; we hinted earlier at the mechanical translation rules to get $\varphi^{\mathbb{N}}$ into official form:

$$(\forall x \geq 0)\psi(x) \text{ becomes } (\forall x)[x \geq 0 \to \psi^{\mathbb{N}}(x)]$$
$$(\exists x \geq 0)\psi(x) \text{ becomes } (\exists x)[x \geq 0 \wedge \psi^{\mathbb{N}}(x)]$$

The translation rules have to be applied recursively, as indicated. Example: $(\exists y)(\forall x)[x \geq y]$ (true in $\mathbb{N}$, false in $\mathbb{Z}$) becomes

$$(\exists y)[y \geq 0 \wedge (\forall x)[x \geq 0 \to y \geq x]]$$

where I haven't yet eliminated *all* the vernacular, but it should be clear how to do that. So the relativized formula $\varphi^{\mathbb{N}}$ is satisfied by $\mathbb{Z}$. Informally, we say that $\varphi^{\mathbb{N}}$ "talks only about $\mathbb{N}$".

We can pull this trick anytime we have a structure $\mathbf{S}$ and a subclass of the universe of $\mathbf{S}$, say $A \subseteq S$, provided A can be defined by a formula of the language of $\mathbf{S}$. That is, provided that $A = \{s \in S : \mathbf{S} \models \alpha(s)\}$ for some formula α.

A picky point: what if A is not closed under the functions of S? For example, what if we added a subtraction symbol to the language of INTEGERS? We could have a formula φ in $\mathcal{L}$(INTEGERS) where the relativized formula $\varphi^{\mathbb{N}}$ talks (indirectly) about negative integers. The usual workaround is to replace the function symbols with predicates, like we did with GROUP, replacing $\cdot$ with EQ.

Even without the workaround, the relativized formula $\varphi^{\mathbb{N}}$ (or more generally φ^{α}) still makes sense as a formula in the language of the larger structure $\mathbf{S}$. The point of relativization is the equivalence of these semantic statements:

$$\mathbf{S} \models \varphi^{\alpha} \quad \text{if and only if} \quad \mathbf{A} \models \varphi$$

where $\mathbf{A}$ is "$\mathbf{S}$ restricted to A". Now, if A is not closed under all the functions of $\mathbf{S}$, then we can't even define that restriction. (First-order logic

doesn't allow function symbols to be interpreted as partial functions.) The workaround works around that problem.

Although relativization is motivated by sematics, we can describe the transformation

$$\varphi \text{ becomes } \varphi^\alpha$$

purely syntactically, i.e., mechanically.

For the theories INTEGERS and PA, we expect more. Say φ belongs to $\mathcal{L}(\text{PA})$. Well, as a string of symbols, φ also belongs to $\mathcal{L}(\text{INTEGERS})$. So we can relativize it to get $\varphi^{\mathbb{N}}$. We expect that $\text{PA} \vdash \varphi$ if and only if $\text{INTEGERS} \vdash \varphi^{\mathbb{N}}$. This is true, provided INTEGERS is set up correctly. Indeed, any proof in PA can be translated mechanically to a proof in INTEGERS.

Still more generally, suppose we have theories $\mathcal{S}$ and $\mathcal{T}$. Suppose we have a mechanical way of translating formulas in the language of $\mathcal{S}$ into the language of $\mathcal{T}$, say:

$$\begin{gathered}\mathcal{L}(\mathcal{S}) \to \mathcal{L}(\mathcal{T}) \\ \varphi \mapsto \ulcorner\varphi\urcorner\end{gathered}$$

such that if $\mathcal{S} \vdash \varphi$ then $\mathcal{T} \vdash \ulcorner\varphi\urcorner$. In that case, $\mathcal{T}$ in some sense can "emulate" $\mathcal{S}$. (Sort of like a Linux system emulating Windows.) If $\mathcal{S}$ is inconsistent, the $\mathcal{T}$ must also be inconsistent. (That's assuming that if φ is of the form $\psi \wedge \neg\psi$, ψ closed, then so is $\ulcorner\varphi\urcorner$. Or at least $\ulcorner\varphi\urcorner$ is some sort of inconsistency.) This would be like the Linux system crashing if the Windows system it was emulating crashed. (I understand real emulators try to avoid this.)

Example: PA can be emulated in ZF. We translate 0 to $\varnothing$, 1 to $\{\varnothing\}$, $x + y$ to ordinal addition, $x \cdot y$ to ordinal multiplication, and relativize everything to ω. If you dive down into the weeds, the translation is complicated, since $+$ and $\cdot$ are basic symbols in $\mathcal{L}(\text{PA})$ and ordinal addition and multiplication are elaborate constructs in ZF. Still, we can show that if $\text{PA} \vdash \varphi$, then $\text{ZF} \vdash \ulcorner\varphi\urcorner$.

Actually, ZF is more powerful than PA, even when talking only about ω. That is, there are formulas φ for which

$$\mathrm{ZF} \vdash \ulcorner \varphi \urcorner$$

but

$$\mathrm{PA} \nvdash \varphi$$

Perfect example: the Gödel formula Con(PA) that says that PA is consistent. Its translation $\ulcorner \mathrm{Con(PA)} \urcorner$ is a theorem of ZF.

When this sort of thing doesn't happen, we say $\mathcal{T}$ is a **conservative extension** of $\mathcal{S}$. Example: in NBG, we have a formula $\mathrm{set}(x)$ that says that x is a set. Then

$$\mathrm{ZF} \vdash \varphi \text{ if and only if } \mathrm{NBG} \vdash \varphi^{\mathrm{set}}$$

So NBG is a conservative extension of ZF: any theorem of NBG that "talks only about sets" is already a theorem of ZF.

Non-Euclidean geometry furnishes another example of emulation, the first one historically. Initially non-Euclidean geometry had trouble finding acceptance. Opposition to it finally crumbled when it was shown that Euclidean geometry can emulate non-Euclidean geometry: if we say that the non-Euclidean plane is really the interior of a disk, that non-Euclidean lines are really certain circular arcs, that non-Euclidean distance is really a certain function of the Euclidean distance, then it turns out that the axioms of non-Euclidean geometry hold, thus reinterpreted (or emulated). (Picky point: there's more than one non-Euclidean geometry. This one is so-called hyperbolic geometry.)

6 Inner Models

Gödel's method of inner models works as follows. We have a theory $\mathcal{T}$ and a formula φ in $\mathcal{L}(\mathcal{T})$. Question: is $\mathcal{T} + \varphi$ consistent? Suppose we can find

a formula $\alpha(x)$ in $\mathcal{L}(\mathcal{T})$ such that this holds:

$$\begin{aligned} \text{for all } \psi \in \mathcal{T}, \quad & \mathcal{T} \vdash \psi^\alpha \\ \text{and also} \quad & \mathcal{T} \vdash \varphi^\alpha \end{aligned}$$

Let's say we had a proof of a contradiction in $\mathcal{T} + \varphi$. Go through and relativize everything with α. Now we have a proof of a contradiction in $\mathcal{T}$: all the relativized axioms of $\mathcal{T} + \varphi$ can be proved in $\mathcal{T}$, and it turns out that relativization preserves the logical axioms and rules of inference. (Picky point: we need $(\exists x)\alpha(x)$ to hold too.)

This is a purely syntactic argument, but it's motivated by a semantic one. Let's say $\mathbf{T}$ is a model for $\mathcal{T}$. We now consider the substructure selected by $\alpha(x)$, call it $\mathbf{A}$. It's a model of $\mathcal{T} + \varphi$! That's because each formula $\psi \in \mathcal{T}$, when interpreted as speaking about $\mathbf{A}$, is equivalent to ψ^α interpreted in $\mathbf{T}$. As we saw:

$$\mathbf{A} \models \psi \quad \text{if and only if} \quad \mathbf{T} \models \psi^\alpha$$

Likewise for φ. We've found a model of $\mathcal{T} + \varphi$ sitting inside a model of $\mathcal{T}$. $\mathbf{A}$ is called an **inner model**, for obvious reasons.

This is what Gödel did with L. Of course the hard part is proving the relativizations:

$$\begin{aligned} \text{for all } \psi \in \mathrm{ZF}, \quad & \mathrm{ZF} \vdash \psi^L \\ \text{and also} \quad & \mathrm{ZF} \vdash \varphi^L \end{aligned}$$

Here, φ is $V = L$, "All sets are constructible". So we have Gödel's celebrated consistency result: $\mathrm{Con(ZF)} \to \mathrm{Con(ZF}{+}V = L)$. (People write ZFL for ZF$+V = L$.) Gödel also showed that ZFL$\vdash$AC and ZFL$\vdash$GCH, so we have relative consistency for these too.

Here's a trivial example of the method of inner models. Let GROUP be the first-order theory of groups, and let ABELIAN be the axiom $x \cdot y = y \cdot x$. Any model of GROUP (i.e., any group) has a model of GROUP+ABELIAN sitting

inside of it, namely its center. The formula

$$\zeta(x) \Leftrightarrow (\forall y)[x \cdot y = y \cdot x]$$

selects the center of the group. (Picky point: we can't let the $(\forall y)$ be implicit, since we need $\zeta(x)$ to define a unary relation.) Within GROUP, we can prove that the center of a group is an abelian group. It's not *totally* trivial that the center of a group is even a group, i.e., that it's closed under the group operation. Anyway, this argument shows the relative consistency result

$$\text{Con}(\textsc{Group}) \rightarrow \text{Con}(\textsc{Group+abelian})$$

admittedly a trivial result, but it illustrates the method.

Finally, let's note an important feature of the inclusion $L \subseteq V$. Is it a proper inclusion, i.e., are there any non-constructible sets? Gödel thought so. So do most set-theorists who believe the question has meaning. For a formalist, the only question that has meaning is, what can you prove? Well, if we did have ZF $\vdash V \neq L$, that would mean that ZFL was inconsistent. By Gödel's relative consistency result, that can happen only if ZF itself is inconsistent!

Cohen showed that ZF+$V \neq L$ is also consistent (if ZF is), so just like AC and GCH, whether $V = L$ cannot be settled by the axioms of ZF. For a formalist, that's the end of the story. For a platonist—some one who believes that the universe of set theory "really exists"—the question still has meaning. (Your platonism has to be at least moderately strong: you could believe that a multiverse of sets "really exists", with $V = L$ true in some universes and not in others.)

For what it's worth, the consensus among set theorists of the platonist persuasion seems to be that AC is true, GCH is false, and $V = L$ is also false.

7 Model Theory

Here are the big, basic theorems of model theory:

Downward Löwenheim-Skolem Theorem: Any model $\mathbf{N}$ of a countable theory $\mathcal{T}$ has a countable elementary submodel. "Countable" theory means that its language contains only countably many symbols. "Elementary submodel" means, informally: statements that refer only to elements of $\mathbf{M}$ hold above in $\mathbf{N}$ if and only if they hold below in $\mathbf{M}$. I'll give a more precise version later.

Gödel's Completeness Theorem: Every consistent theory has a model.

Compactness Theorem: If every finite subtheory of $\mathcal{T}$ has a model, then $\mathcal{T}$ has a model. "Finite subtheory" means a finite number of nonlogical axioms.

I'll say more about all these results below. Since I don't want to duplicate textbooks, I won't give formal proofs. But I'll often talk "around" the proofs, to a degree where you might find it easy to fill in the details. (Or not.)

7.1 Elementary Equivalence and Elementary Submodel

In preparation for the Löwenheim-Skolem Theorems, we define "elementary substructure". Let $\mathcal{L}$ be a first order language and let $\mathbf{M}$ and $\mathbf{N}$ be structures for $\mathcal{L}$. Also let $\mathbf{M}$ be a substructure of $\mathbf{N}$, i.e., $M \subseteq N$ and the functions and relations of $\mathbf{M}$ are restrictions of their versions in in $\mathbf{N}$. Let's write $\mathbf{M} \sqsubseteq \mathbf{N}$ for substructure.

We say $\mathbf{M}$ is an **elementary substructure** of $\mathbf{N}$ ($\mathbf{M} \preceq \mathbf{N}$) if for any $\vec{a}$

contained in M, and any $\psi(\vec{x})$ in $\mathcal{L}$,

$$\mathbf{M} \models \psi(\vec{a}) \Leftrightarrow \mathbf{N} \models \psi(\vec{a})$$

"As above, so below."[3] We don't actually need "if and only if": we could just say "if", and then look at $\neg\psi(\vec{a})$ for the other direction.

In words: *assertions* about $\mathbf{M}$ hold in $\mathbf{M}$ iff they hold in $\mathbf{N}$. Recall from §3.3 that *assertion* is my term for "closed formula with names allowed", i.e., closed formula in $\mathcal{L}_M$.

Elementary substructure is stronger than elementary equivalence: $\mathbf{M}$ is **elementarily equivalent** to $\mathbf{N}$ when for any closed ψ in $\mathcal{L}$,

$$\mathbf{M} \models \psi \Leftrightarrow \mathbf{N} \models \psi$$

In words: *pure sentences* hold in $\mathbf{M}$ iff they hold in $\mathbf{N}$. Here we don't demand any relation between M and N, or between their functions and relations. So the key difference is that *elementary substructure* allows names for arbitrary elements.

If $\mathbf{N}$ is a a model of a theory $\mathcal{T}$ (in the language $\mathcal{L}$), then any elementary substructure $\mathbf{M}$ is also a model of $\mathcal{T}$, obviously. So people say "elementary submodel" in this case.

The language of linear orderings, $\mathcal{L}(\mathrm{LO})$ (see 4.2) provides a slew of informative examples. First let's use it to illustrate the difference between elementary submodel and elementary equivalence. Look at the closed intervals $[0,1] \subseteq [0,2]$. They are elementarily equivalent, indeed, order isomorphic. But $[0,1]$ is not an elementary substructure of $[0,2]$, because $[0,1] \models (\forall x)(x \leq 1)$ while $[0,2] \models \neg(\forall x)(x \leq 1)$. The name for 1 changes the game.

The theory DLO (dense linear orderings without endpoints) is complete, as was mentioned in 4.2. So any two models of DLO (DLOs, for short)

[3] A quote from the Emerald Tablet of Hermes Trismegistus, a alchemical sacred text.

are elementarily equivalent, since they must satisfy exactly the same pure sentences. As it happens, if $\mathbf{A} \sqsubseteq \mathbf{B}$ are both DLOs, then $\mathbf{A}$ is an elementary submodel of $\mathbf{B}$. Example: $\mathbb{Q} \preceq \mathbb{R}$, even though $\mathbb{Q}$ and $\mathbb{R}$ have different cardinalities.

Another DLO example: let A be the union of all the open intervals $(n, n+\frac{1}{2})$ for all $n \in \mathbb{Z}$, or even the union of all the half-open intervals $[n, n+\frac{1}{2})$. A is an elementary submodel of $\mathbb{R}$—A is *dense*, even though it's not *dense in* $\mathbb{R}$. (To use some antiquated terminology, A is *dense-in-itself*.) In the definition of *elementary submodel*, we are allowed names for all elements of the *contained* structure but not the *containing* structure. In this example, the assertion $(\exists x)[\frac{2}{3} < x < \frac{3}{4}]$ belongs to $\mathcal{L}_{\mathbb{R}}(\mathrm{LO})$, but not to $\mathcal{L}_A(\mathrm{LO})$.

Consider ω and $\omega + \omega$. These are *not* elementarily equivalent, since in the former but not the latter, the smallest element is the only one without an immediate predecessor; writing this formally is not hard. As it happens, ω is an elementary substructure of $\omega + \omega^* + \omega$, i.e., the order type of a copy of $\mathbb{N}$ followed by a copy of $\mathbb{Z}$.

An algebraic example: it turns out that the (countable) field of algebraic numbers is an elementary submodel of the field of complex numbers (with $\mathcal{T}$ being the theory of fields).

7.2 Löwenheim-Skolem Theorems

The **downward Löwenheim-Skolem theorem** says that any structure $\mathbf{N}$ for a countable language $\mathcal{L}$ has a countable elementary substructure $\mathbf{M}$; moreover, one that contains any given countable subset M_0 of N. "Countable language" means countably many symbols, and hence countably many formulas. There is a generalization to infinite cardinalities other than $\aleph_0$, and also an upward Löwenheim-Skolem theorem that gives elementary extensions of arbitrarily large cardinality. We postpone these for now.

The proof of the downward Löwenheim-Skolem theorem vaguely resembles the construction of the field of algebraic numbers. There, we just keep adding roots of polynomial equations (plucked from $\mathbb{C}$) to our base field $\mathbb{Q}$. Special properties of polynomials make the construction especially pleasant: it's enough to throw in all roots of all polynomial equations with rational coefficients in one fell swoop—then you're done. For the Löwenheim-Skolem theorem, we start with $M = M_0$. We keep adding "roots" to M, i.e., elements c (plucked from N) satisfying $\mathbf{N} \models \varphi(c)$ for various φ's. We don't have the nice polynomial properties, so we have to iterate ω times. But the basic idea is the same.

The language $\mathcal{L}(\mathrm{LO})$ (linear orderings) furnishes a particularly transparent example. We let $\mathbf{N}$ be $(\mathbb{R}, <)$, and we let M_0 be $\mathbb{Z} \subseteq \mathbb{R}$. We construct a countable M with $\mathbb{Z} \subseteq M \preceq \mathbb{R}$. Now, $\mathbb{R}$ satisfies the density condition: $(\forall x < z)(\exists y)[x < y < z]$. Stage 0: $M_0 = \mathbb{Z}$. Stage 1: for any integer i, pick a real between i and $i+1$ and add it; let M_1 be M_0 plus all these added elements. (The element added between i and $i+1$ is called a **witness** to $(\exists y)[i < y < i+1]$.) Stage $n+1$: for any two adjacent elements of M_n, pick a real between them and add it; let M_{n+1} be M_n plus all these added elements. (Again, the added elements "witness" assertions $(\exists y)[r < y < s]$ where r and s are adjacent elements of M_n.) Finally, we let $M = \bigcup_n M_n$. Then $M \preceq \mathbb{R}$. (In particular, M is a DLO; as we saw in §4.2, there's no guarantee that M is dense in $\mathbb{R}$.)

It doesn't matter which elements of $\mathbb{R}$ are chosen as witnesses at each step; for concreteness, let's say we always choose the average $(r+s)/2$ of the two names. In that case, M will be the set of dyadic rationals (i.e., denominator a power of 2).

This DLO example illustrates some aspects of the general argument. Note that none of the M_n's are DLOs, just the final union. In the general case, we obtain M as a union of increasing sequence M_n. We obtain M_{n+1} by adding "witnesses" for all assertions of the form

$$(\exists x)\varphi(x, \vec{c}) \tag{1}$$

where all the entries in $\vec{c}$ are elements of M_n. In the general case, we have to appeal to the axiom of choice to choose a single witness for each such assertion. None of the M_n's will generally be elementary substructures, just the final union M.

In one respect, the $\mathbb{Z} \subseteq \mathbb{R}$ DLO example is *too* simple. First, in the general case, we consider all assertions of the form (1). In our DLO argument, we considered, at stage n, only assertions of the specialized form

$$(\exists x)[c < x < d] \qquad c, d \in M_n, c < d, c \text{ and } d \text{ adjacent in } M_n \qquad (2)$$

Such assertions are of the form $(\exists x)\varphi(x, \vec{c})$ where the inner φ is quantifier-free. Not only that, but adding witnesses achieved the desired result at the very next stage: (2) holds in M_{n+1}. In general that might not happen.

For the general case, all we are sure of is

$$\mathbf{M} \models \psi(\vec{c}) \Leftrightarrow \mathbf{N} \models \psi(\vec{c}), \qquad \vec{c} \subseteq M$$

for all such $\psi(\vec{c})$. The proof is by induction on the complexity of the assertions, and is just turning the crank except for one case: if (1) holds in $\mathbf{N}$, then it holds in $\mathbf{M}$. Even that's not difficult: since $M = \bigcup M_n$, there is an M_n that contains all the entries in $\vec{c}$. So a witness for it is added to M_{n+1}, say b. Since $\mathbf{N} \models \varphi(b, \vec{c})$, by induction $\mathbf{M} \models \varphi(b, \vec{c})$ and so (1) holds in $\mathbf{M}$.

The Löwenheim-Skolem theorems lead to Skolem's paradox—actually a whole family of paradoxes. General idea: we write down a first order theory $\mathcal{T}$ that is *meant* to define some uncountable structure, like the real numbers. Standard result from analysis: $\mathbb{R}$ is a complete ordered field, and all complete ordered fields are isomorphic. OK, write down the first-order theory for this, in a rich language, not just the one we had for real-closed fields. (Maybe throw in predicates for integers, sets of reals, functions, etc.) By downward LS, $\mathbb{R}$ contains a countable model of the theory. What gives? Answer: first-order logic can never capture the full intended meaning of "complete". All Dedekind cuts, all Cauchy sequences—these escape its grasp.

You can formalize the complete ordered field axioms in ZFC, and prove that all complete ordered fields are isomorphic. The Löwenheim-Skolem theorems can be formalized in ZFC for the case where the models have "small" universes (i.e., the universes are sets). But now we have a particularly striking version of Skolem's paradox. We can carry out the "proof" of downward LS informally *outside* ZFC on the universe V of ZFC itself! Result: a countable model $(M, \in)$ of ZFC, with the same $\in$ relation as the real one in V. M is a countable set that "looks like" the whole universe!

Suppose we drop the countability assumptions; we assume the language $\mathcal{L}$ and the initial subset M_0 both have cardinality at most $\mathfrak{m}$. (The cardinality of a language is defined as the cardinality of the set of all its formulas.) The argument sketched above still works, and doing some simple transfinite arithmetic, we conclude that the final submodel M has cardinality at most $\mathfrak{m}$ too. This is the generalized downward LS theorem.

The upward LS theorem says that if $\mathbf{M}$ is a structure for a language $\mathcal{L}$, then $\mathbf{M}$ has elementary extensions for all cardinalities greater than or equal to the cardinality of $\mathcal{L}_M$. The proof uses a different technique—it's a consequence of the Gödel Completeness Theorem (generalized to all cardinalities). The upwards LS theorem begets its own variant of the Skolem paradox: the existence of uncountable models of Peano arithmetic, or even of true arithmetic. I won't discuss it further.

7.3 Gödel's Completeness Theorem

The Gödel's completeness theorem says that any consistent first-order theory has a model (in fact, a countable model, if the language is countable).

The name "completeness theorem" comes from this consequence:

> Suppose that for all models $\mathbf{M}$ of $\mathcal{T}$, $\mathbf{M} \models \varphi$. Then $\mathcal{T} \vdash \varphi$, i.e., φ is a theorem of $\mathcal{T}$.

In other words, the rules of first-order logic are "complete": if being a model of $\mathcal{T}$ guarantees that you also satisfy φ, then this implication can be proved formally.

The consequence comes about because of this fact:

> $\mathcal{T} + \neg\varphi$ is *inconsistent* if and only if $\mathcal{T} \vdash \varphi$.

Basically, this describes a proof by contradiction: you suppose $\neg\varphi$, obtain a contradiction, and conclude φ. The rules of (classical) logic were designed to make this work.

So suppose that $\mathbf{M} \models \varphi$ for all models $\mathbf{M}$ of $\mathcal{T}$. That means there are *no* models of $\mathcal{T} + \neg\varphi$. So by the completeness theorem (in contrapositive form), $\mathcal{T} + \neg\varphi$ must be inconsistent. So we have $\mathcal{T} \vdash \varphi$.

7.4 Compactness Theorem

This is a simple consequence of the completeness theorem, but it's used constantly in model theory.

Important but obvious fact:

> Only finitely many postulates can appear in a proof.

Corollary:

> If a theory $\mathcal{T}$ is inconsistent, then a finite subtheory of T is already inconsistent.

Or:

If every finite subtheory of $\mathcal{T}$ is consistent, then $\mathcal{T}$ is consistent.

First example: nonstandard arithmetic. Let n be a new constant added to the language of Peano arithmetic. Add the infinite set of postulates:

$$\begin{aligned} n &> 1 \\ n &> 1+1 \\ n &> (1+1)+1 \\ n &> ((1+1)+1)+1 \\ &\vdots \end{aligned}$$

Any finite subset is consistent, so there is a model with an "infinite integer" n. Note that this works even with all the postulates of true arithmetic thrown in.

Second example: quasi-finite fields. Like the first example, but use p instead of n and add a postulate that p is prime. Now, the field $\mathbb{Z}_p$ (integers mod p) is well-known to algebraists as a finite field. It's no big deal to translate statements about $\mathbb{Z}_p$ into statements in the language of PA. You can piggyback on this to define an actual structure $\mathbb{Z}_p$ when p is an "infinite prime" in a nonstandard model of PA (or TA). Such a $\mathbb{Z}_p$ acts as a kind of "quasi-finite field". Regarded from "outside", it has characteristic zero. From "inside", it has characteristic p.

Third example: Robinson's nonstandard analysis. Abraham Robinson started with a particularly rich language for the real numbers, one that allowed discussion of natural numbers, sets of reals, functions from reals to reals, etc. Using the same trick as above, he got a model with an "infinite integer". Since this is a field, we also get "infinitesimals", like $1/n$, $1/n^2$, etc. Now, Leibniz and Newton used infinitesimal freely, and physicists never gave them up. Robinson show how all those arguments could be rigorously justified in the new framework. He even used the machinery to settle an open question in functional analysis (the "polynomially compact operator problem").

(The idea was to treat linear operators as given by "infinite matrices".) Analysts were horrified by this and promptly figured out how to rederive the result by more traditional means.

Fourth example: the infinite four-color theorem. Let G be a planar graph with infinitely many vertices. Add a constant a_i to the language for every vertex. Add postulates like

$$a_i \sim a_j$$

whenever the corresponding vertices have a connecting edge, and

$$\neg(a_i \sim a_j)$$

whenever the corresponding vertices don't have a connecting edge. Any model of this augmented theory will contain a subgraph isomorphic to G.

We saw in §4.1 how to augment the theory GRAPH with new predicate symbols for the four colors, and some new axioms. A model of this theory, plus all the $a_i \sim a_j$ and $\neg(a_i \sim a_j)$ axioms, will contain a four-colored graph isomorphic to G.

So, *if* this doubly-augmented theory has a model, *then* G can be four-colored. Now we apply compactness: The doubly-augmented theory has a model if every finite subtheory does. A finite subtheory mentions only finitely many vertices. Looking just at those vertices, we have a finite subgraph G_0 of G. Since G_0 is a finite planar graph, the four color theorem applies to it, and we have a model of the finite subtheory.

8 Definability in Peano Arithmetic

As mentioned in §4.3, PA is surprisingly expressive. We will need the specifics several times in later sections, especially §§9.1, 10.1, 10.2, and 11.2. The material can be a little dry. On the positive side, §8.4 shows off some clever coding tricks.

8.1 Functions and Relations

We say a relation R on $\mathbb{N}^k$ is **arithmetically definable** (or just **arithmetic**) if there is a formula $\varphi(\vec{x})$ in $\mathcal{L}(\text{PA})$ such that

$$R(\vec{a}) \text{ holds iff } \mathbb{N} \models \varphi(\vec{a}), \text{ for all } \vec{a} \in \mathbb{N}^k$$

(See §10.2.)

We say a function $f : \mathbb{N}^k \to \mathbb{N}$ is **arithmetically definable** (or **arithmetic**) if its graph is arithmetic, i.e.,

$$f(\vec{a}) = b \text{ holds iff } \mathbb{N} \models \varphi(\vec{a}, b), \text{ for all } \vec{a} \in \mathbb{N}^k, b \in \mathbb{N}$$

for some formula $\varphi(\vec{x}, y)$ in $\mathcal{L}(\text{PA})$.

Oftentimes, we can prove that the formula defines a function:

$$\text{PA} \vdash (\forall \vec{x})(\exists! y)\varphi(\vec{x}, y)$$

but this is not traditionally part of the definition.

Logicians have developed a whole machinery of "extension by definition" for this sort of situation. As a couple of simple examples, the relation $x \leq y$ is defined by $(\exists z)x + z = y$; the function $(x, y) \mapsto x^y$ can be defined by a formula, say $\exp(x, y, z)$, as we will see in §8.4. (One has to decide on a value for 0^0; we choose $0^0 = 1$.)

In such cases, you can expand the language, to get what's called a *conservative extension*. For example, let $(N, +, \cdot, 0, 1)$ be a structure for $\mathcal{L}(\text{PA})$. The formulas for $x \leq y$ and x^y tell us exactly how to "expand" this structure to a structure $(N, +, \cdot, 0, 1, \leq, x^y)$, by uniquely defining the new relation $\leq$ and the new function x^y. That's at the semantic level. At the purely syntactic "theory" level, let $\mathcal{L}(\text{PA}^*)$ be the expanded language, and in that language add these two new axioms to PA to get PA*:

$$x \leq y \leftrightarrow (\exists z)x + z = y$$
$$x^y = z \leftrightarrow \exp(x, y, z)$$

Any formula in the expanded language can be mechanically translated into the original language; any closed formula in the expanded language is provable in PA* if and only if translated formula is provable in PA. This last claim relies on the fact that PA can prove the "functionness" of exp, i.e.,

$$\mathrm{PA} \vdash (\forall x, y)(\exists! z)\, \exp(x, y, z)$$

One says that PA* is a **conservative extension** of PA.

I've generally skirted this nitty-gritty by glibly saying "vernacular". I thought I should at least mention how a more scrupulous treatment would look.

It turns out that all *recursive* relations and functions (aka *computable*) are arithmetic. I won't go so far as to prove this, or even sketch a proof, but I will present the key techniques that go into the proof. For now, just lodge this as a fact, as an aid to intuition.

8.2 Recursive Tupling Functions

As you know, we have a 1–1 correspondence $\mathbb{N} \times \mathbb{N} \leftrightarrow \mathbb{N}$. Lots of them! We'll pick one in a moment; we'll denote it by

$$(x, y) \leftrightarrow \langle x, y \rangle$$

Here $(x, y) \in \mathbb{N} \times \mathbb{N}$ is an ordered pair, and $\langle x, y \rangle \in \mathbb{N}$ is the number coding it.

We want $\langle \cdot, \cdot \rangle$ to be arithmetically definable. The following formula does the trick:

$$\langle x, y \rangle = \frac{(x + y)(x + y + 1)}{2} + y$$

It's not hard to see where it comes from. If you list the lattice points in $\mathbb{N} \times \mathbb{N}$ by "cross diagonals" (i.e., (0,0), (1,0), (0,1), (1,1), ...), then the

predecessors of (x, y) consist of a triangle, plus a "tail" lying along the diagonal. Count them up! You get the above formula. So $\langle \cdot, \cdot \rangle$ is not just arithmetic, but recursive.

You can iterate this to get recursive k-tupling correspondences $\mathbb{N}^k \leftrightarrow \mathbb{N}$ for every k. For example,

$$\langle x, y, z \rangle = \langle \langle x, y \rangle, z \rangle$$

Nice enough, but as we will see in §8.3, we really need a *single* mechanism to handle sequences of arbitrary finite length.

However, the tupling functions do allow us to translate formulas like this:

$$(\forall \vec{x})(\exists \vec{y}) \ldots$$

into this:

$$(\forall x)(\exists y) \ldots\ldots$$

See §8.5 for details.

8.3 Recursion

The primary tool for defining arithmetic functions is recursion. Here's the simplest form. Suppose we have a function $f : \mathbb{N} \to \mathbb{N}$, and let a be any number. Define F by

$$F(0) = a$$
$$F(n+1) = f(F(n))$$

If f is arithmetic, does that make F arithmetic? Here are two ways we can write a formula for $F(x) = y$, *provided* we can quantify over arbitrary finite

sequences $a_0, \ldots, a_x$:

$$\begin{aligned} F(x) = y \Leftrightarrow (\exists a_0, \ldots, a_x)[& \\ & a_0 = a \\ & \wedge (\forall z < x)(a_{z+1} = f(a_z)) \\ & \wedge y = a_x] \end{aligned}$$

and

$$\begin{aligned} F(x) = y \Leftrightarrow (\forall a_0, \ldots, a_x)[& \\ & a_0 = a \\ & \wedge (\forall z < x)(a_{z+1} = f(a_z)) \\ & \rightarrow y = a_x] \end{aligned}$$

So we have a choice of $\exists$ or $\forall$; this will be important later. In §8.4 we'll show how to handle arbitrary finite sequences in PA.

More complicated forms of recursion? No problem. For example, say $f(x, \vec{y}, z)$ is a function of $n+2$ variables and $g(\vec{y})$ of n variables. Define

$$\begin{aligned} F(0, \vec{y}) &= g(\vec{y}) \\ F(n+1, \vec{y}) &= f(F(n, \vec{y}), \vec{y}, n) \end{aligned}$$

So we're allowing the new value to depend on a bunch of parameters $\vec{y}$, on the argument n, and of course on the previous value $F(n, \vec{y})$. This scheme is called **primitive recursion**. It's handled the same way as the simpler case.

With this in our toolbox, we can define any so-called *primitive recursive* function. That's any function you get by starting with a few very basic ones (projection, constant functions, successor) and applying composition and primitive recursion as many times as you like. So-called "course of values" recursions, where the new value depends on all values previously computed, also are available, since we can code the list of previous values using the

techniques of §8.4. Sets and relations can be coded via functions taking the values 0 and 1; then we can use such relations in boolean conditionals (or more generally "definition by cases") when defining new functions.

Loosely speaking, primitive recursive functions coincide with the computable functions where you can "clearly see" that the computations always terminate. To get the full panoply of **recursive functions**, we add the so-called μ-operator. This defines a *partial* function $\vec{x} \mapsto y$, like so:

$$(\mu y)\varphi(y, \vec{x}) = \begin{array}{l} \text{the least } y \text{ such that } \varphi(y, \vec{x}) \\ \text{undefined if there is no such } y \end{array}$$

It's easy to express this in the language of PA:

$$(\mu y)\varphi(y, \vec{x}) = u \Leftrightarrow \varphi(u, \vec{x}) \wedge (\forall z < u)\neg\varphi(z, \vec{x})$$

Any function defined by applying the μ-operator to a primitive recursive relation is a **partial recursive function**. If a partial recursive function just happens to be total, it's a **recursive function**. But we can't (computably) tell in general if a partial recursive function is total—that's the Halting Problem on steroids.

The "sequence trick" and the μ-operator together tell us that any partial recursive function is arithmetically definable. But these techniques go beyond this: if f and g are arithmetic, then so is the F defined by the primitive recursion scheme, even if f and g are horribly, hideously uncomputable. We'll explore this a little more in §8.5.

8.4 Finite Sequences

As we've just seen in §8.3, we need a way to code finite sequences of arbitrary length as single numbers. Gödel gave one technique: code $(r_1, \ldots, r_k)$ as $p_1^{r_1} \cdots p_k^{r_k}$, where p_i is the i-th prime. But this works only once we have a

definition of exponentiation, and of the function $i \mapsto p_i$. Both these yield easily to recursion, but that requires a coding of finite sequences!

Gödel broke the vicious circle with the Chinese remainder theorem. This says that given any sequence of positive integers $d_1, \ldots, d_k$, all pairwise coprime, and any sequence $r_1, \ldots, r_k$ with $0 \leq r_i < d_i$ for $i = 1, \ldots, k$, there is an a such that $a\%d_i = r_i$ for all i. Here $\%$ is the remainder function. We call the d_i's *divisors* and the r_i's *remainders*.

The Chinese remainder theorem is actually stronger: it says that there is a unique such a satisfying $0 \leq a < d_1 \cdots d_k$. Although we won't need the stronger version, it falls right out of the easiest proof of the theorem. Let $d = d_1 \cdots d_k$. Write $[d]$ for the set $\{0, \ldots, d-1\}$, likewise for $[d_i]$. We have a mapping $[d] \to [d_1] \times \ldots \times [d_k]$ defined by $a \mapsto (a\%d_1, \ldots, a\%d_k)$. The mapping is injective because the d_i's are pairwise coprime: if $a\%d_i = a'\%d_i$ for all i, then $d_i|(a - a')$ for all i and so $d|(a - a')$. Since the domain and codomain both have d elements, the mapping is surjective. qed.

So if we are given a finite sequence $(r_1, \ldots, r_k)$, we just have to find divisors d_i such that they are pairwise coprime and $r_i < d_i$ for all i. Well, the sequence of divisors has to be "orderly", in the sense that if we had to specify all the d_i's individually, we wouldn't have gained anything.

Gödel chose the arithmetic progression $d_i = bi + 1$, for a suitable b. Let's see what conditions we must impose on b. First if b is greater than all the r_i's, we'll have $r_i < d_i$ for all i. Next, if a prime p divides some d_i, it can't divide b. If p divides d_i and d_j for some i and j, it must divide $d_i - d_j = b(i - j)$, and since p doesn't divide b, it would have to divide $i - j$. If we can compel p to be greater than k, this will imply $i = j$, making the divisors pairwise coprime. If $b = k!$ then p not dividing b will imply $p > k$. If $k!$ isn't greater than all the r_i, then we let b be a sufficiently large multiple of $k!$.

Now define

$$\beta(a, b, i) = a\%(bi + 1)$$

By the Chinese remainder theorem and the reasoning above, for any finite sequence $(r_1, \ldots, r_k)$ there is an a and a b such that $\beta(a, b, i) = r_i$ for $i = 1, \ldots, k$. That's all we need to replace a quantifier $\exists(r_1, \ldots, r_k)$ with $\exists a \exists b$. (Formally representing the remainder function is a piece of cake.) Proving that all primitive recursive functions are definable in PA is now a walk in the park. Once you have all the primitive recursive functions, you get all the recursive functions with just one more existential quantifier; indeed, all the partial recursive functions. (As a side note, we can use the recursive pairing function to give the β function two arguments instead of three.)

Now for a very different approach, based on binary notation.[4] We build up to finite sequences through a chain of relations and functions.

Pow2(x), x is a power of 2.

$$(\forall y)[\text{Prime}(y) \wedge y | x \rightarrow y = 2]$$

y=RoundUp2(x), y is x "rounded up" to the next power of 2. I.e., $2^{m-1} \leq x < 2^m = y$. (If x is a power of 2, we still round it up to the next power of 2.) We ignore the case $x = 0$ (see below).

$$(\exists u)[\text{Pow2}(u) \wedge u \leq x < y \wedge y = 2 \cdot u]$$

z=Concat(x,y), z is x concatenated with y (when all three are written in binary).

$$z = x \cdot \text{RoundUp2}(y) + y$$

Strictly speaking, this doesn't work if x or y is 0. There is an easy way to resolve this glitch: see below.

[4]This solution comes from the exercises to §3.1 of Kaye [4], who credits the ideas to Quine [8] and Smullyan [11].

MemberOf(x,y), x is a member of a finite set represented by y. This takes a bit more work. Say we have a finite set $\{a_1, \ldots, a_k\}$. (Assume none of the a_i's is 0: see below.) Write each a_i in binary, and separate them by strings of 1's, say $s = 11\ldots1$. So we want $y = a_1 s a_2 s \ldots s a_k$ (the right side a concatenation of binaries, of course). To avoid parsing ambiguity, we make s longer than all of the a_i's. So if x is one of the a_i's, we can write y as $usxsv$. Obviously the five-place concatentation can be defined using Concat, and s is a string of 1's iff it's a power of 2 minus 1. So:

$$(\exists u, s, v)[x < s \wedge \text{Pow2m1}(s) \wedge y = \text{Concat}(u, s, x, s, v)]$$

x=Seq(y,i), x is the i-th item in the sequence represented by y. We simply use a recursive binary pairing function and represent $(r_1, \ldots, r_k)$ as $\{\langle 1, r_1\rangle, \ldots, \langle k, r_k\rangle\}$. Then $x = \text{Seq}(y, i)$ is equivalent to:

$$\text{MemberOf}(\langle i, x\rangle, y)$$

We resolve the "zero problem" by using a pairing function that doesn't have 0 in its range. So 0's never occur when we are coding finite sets for MemberOf, and we're home free.

With neither approach do we have a unique number representing a finite sequence, but this doesn't matter. Once we have any representation, we can use it to show that all primitive recursive functions are representable in PA; then we can represent $(r_1, \ldots, r_k)$ as $p_1^{r_1} \cdots p_k^{r_k}$ if we need uniqueness.

8.5 The Arithmetic Hierarchy

Arithmetic relations are arranged in a hierarchy according to the complexity of the quantifiers in their definitions.

Basic predicate logic tells us that any formula ψ can be rewritten in **prenex normal form**: i.e., there is a logically equivalent formula with all the quantifiers out front, prefixing a quantifier-free formula. Thus:

$$(\exists \vec{x})(\forall \vec{y}) \ldots \varphi, \text{ or}$$
$$(\forall \vec{x})(\exists \vec{y}) \ldots \varphi$$

As indicated, we have blocks of existential and universal quantifiers; $(\exists \vec{x})$ stands for $(\exists x_1)(\exists x_2) \ldots (\exists x_k)$, as usual. If we have n alternating quantifier blocks, then we have an $\exists_n$ or a $\forall_n$ form, depending on whether we lead off with existential or universal quantifiers respectively.

For PA, we can make some modifications and simplifications. First, we say ψ is Δ_0 if it can be expressed using a string of *bounded quantifiers* in front of a quantifier-free part. For example,

$$(\exists u < x)(\exists v < x)(\forall w < u)[x = \langle u, v \rangle \wedge \ldots]$$

From PA, it follows that any formula is equivalent to one in the form

$$(\exists \vec{x})(\forall \vec{y}) \ldots \varphi, \text{ or}$$
$$(\forall \vec{x})(\exists \vec{y}) \ldots \varphi$$

where φ is Δ_0. If we have n alternating quantifier blocks, then we have a Σ_n or a Π_n form, depending on whether we lead off with existential or universal quantifiers respectively. A Δ_n relation is one that has equivalent Σ_n and Π_n definitions. For convenience, Σ_0 and Π_0 mean the same as Δ_0. Also, we regard Σ_{n-1} and Π_{n-1} as included in $\Sigma_n \cap \Pi_n = \Delta_n$—we should have said "at most n quantifier blocks" above.

Using the n-tupling functions, we can replace a block $\exists \vec{x}$ with a single $\exists x$, and likewise for universal quantifiers. For example, the formula $(\exists u)(\exists v)(\forall w < u) \ldots$ is equivalent to $(\exists x)\delta$, where δ is the example of a Δ_0 formula we gave earlier.

Here's a proposition we won't prove: a function is partial recursive iff it has a Σ_1 definition. Likewise a set (or a relation) is recursively enumerable (r.e.) iff it is Σ_1. The proof isn't terribly hard, given the machinery we've developed, just tedious.

It follows that a relation is recursive iff it is Δ_1 (since the complement of a Σ_n relation is Π_n and vice versa). If a function is recursive, then it's Δ_1; that's because $f(\vec{x}) \neq y$ iff $(\exists z)[z \neq y \wedge f(\vec{x}) = z]$, so $f(\vec{x}) = y$ has the Π_1 definition $(\forall z)[z = y \vee \neg f(\vec{x}) = z]$, and $\neg f(\vec{x}) = z$ has a Π_1 definition.

Remarkably, a function is partial recursive iff it has an $\exists_1$ definition. The proof of this is quite hard—proving it solved a Hilbert problem!

9 Gödel's Incompleteness Theorems

Here are Gödel's famous two incompleteness theorems:

Gödel's First Incompleteness Theorem: If $\mathcal{T}$ is a consistent, recursive, sufficiently rich theory, then there is a closed formula φ in its language which is undecidable, i.e., $\mathcal{T} \nvdash \varphi$ and $\mathcal{T} \nvdash \neg\varphi$.

Gödel's Second Incompleteness Theorem: If $\mathcal{T}$ is a consistent, recursive, sufficiently rich theory, then there is a closed formula $\mathrm{Con}\,\mathcal{T}$, expressing the fact that $\mathcal{T}$ is consistent, which cannot be proved in $\mathcal{T}$.

Why "consistent"? It turns out that under the standard rules of first-order logic, you can prove everything in an inconsistent theory. That is, if $\mathcal{T}$ is inconsistent, then for any φ in the language of $\mathcal{T}$, $\mathcal{T} \vdash \varphi$. (Think of a liar who says, "I'm telling the truth!") So we need to assume that $\mathcal{T}$ is consistent.

Why "recursive"? As we noted before, you can define TA (true arithmetic) as having the same language as PA (Peano arithmetic), and having all true statements about $\mathbb{N}$ as its postulates. For example, the Goldbach conjecture is an axiom of TA iff it's true. Proofs in TA are very short: always length one! Trivially, TA is complete. A theory is **recursive** when you have an algorithm that tells you, for any φ, whether or not φ is an axiom. PA is recursive, TA is not.

Why "sufficiently rich"? Many consistent recursive theories are complete. The theory of dense linear orderings without endpoints is complete. Presburger showed that if you throw out the multiplication symbol but keep the rest of PA, you get a complete theory. (People call this Presburger arithmetic.) We mentioned Tarski's result on real-closed fields in §4.4. It's a similar story for algebraically closed fields. (Exercise: show that if all countable models of $\mathcal{T}$ are isomorphic, and $\mathcal{T}$ is countable, then $\mathcal{T}$ is complete. Use both downward LS and the completeness theorem.)

In Gödel's original formulation, "sufficiently rich" meant "an extension of PA" (or PA itself). Then logicians analyzed exactly what was needed. I won't go into details. But fundamentally, if you can mirror the notion of proof inside the theory, then it's sufficiently rich. Gödel showed how to encode formulas and proofs inside PA, as finite sequences of natural numbers. (It's even easier in ZF.) In particular, we can encode the assertion "PA is consistent" as a number-theoretic formula, which we call Con PA.

The incompleteness theorem plus the completeness theorem begets a new paradox. Assuming Con PA, it follows that PA+$\neg$Con PA is consistent, by the second incompleteness theorem. So by the completeness theorem, PA+$\neg$Con PA has a model! (A self-doubting model, so to speak.) This model contains an element n representing a proof of an inconsistency. But n is nonstandard—it doesn't represent an actual, finite proof when viewed from *outside* the model, in the same way that the nonstandard n's we encountered earlier are not actual elements of $\mathbb{N}$.

Gödel's proof combines two key ideas, plus a grace note. First, Gödel showed how to mirror assertions *about* PA *inside* PA. Second, he showed how to focus this introspection in a single sentence, informally couched as "I am unprovable." Finally, he observed that the consistency of PA is equivalent (in PA) to his self-referring sentence. The next two sections try to bring out the main issues without turning into a textbook.

9.1 Gödel Coding

The idea of Gödel coding has become commonplace and needs no introduction. But an expository problem rears its head—or rather, two heads, like Scylla and Charybdis. The Scylla head: formulas of ungodly length. The Charybdis head: arguments that evaporate into clouds of hand-waving. We face this dilemma with other constructs in logic, e.g., formal definitions of proof or satisfaction.

Here's my approach: write $\ulcorner\varphi\urcorner$ for the Gödel code of a formula φ (a standard convention). So $\ulcorner\varphi\urcorner \in \mathbb{N}$ (or is a set, in the ZF case). Then lean heavily on the $\ulcorner\ \urcorner$ notation. For example, let's say I wanted to state something formally about modus ponens. I'd write $\ulcorner\varphi\urcorner$, $\ulcorner\psi\urcorner$, and $\ulcorner\varphi \to \psi\urcorner$ for the Gödel codes of φ, ψ, and $\varphi \to \psi$. Of course there's some computable function $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ giving the code of $\varphi \to \psi$ from the codes of φ and ψ, but we don't need an explicit symbol for it. The claim

$$\mathrm{PA} \vdash \Big(\mathrm{PA} \vdash \ulcorner\varphi\urcorner \wedge \mathrm{PA} \vdash \ulcorner\varphi \to \psi\urcorner \to \mathrm{PA} \vdash \ulcorner\psi\urcorner\Big)$$

is the "formal vernacular"[5] for

> If φ and $\varphi \to \psi$ are provable in PA, so is ψ, and moreover this fact can be proved formally in PA.

[5] sort of like "business casual"...

Here, PA $\vdash \ulcorner\varphi\urcorner$ is vernacular for the formalization of PA $\vdash \varphi$, which is a formula of PA in which the code for φ appears. I'll still need to convince you (or you'll need to convince yourself) that PA $\vdash \varphi$ *can* be formalized. Hand-waving remains indispensable.

Another example: if $\varphi(x)$ is a formula with one free variable and $n \in \mathbb{N}$, then $\varphi(\bar{n})$ is a closed formula; here $\bar{n}$ is the so-called **numeral** for n, i.e., the closed term $1 + \cdots + 1$ with n 1's (or 0 if $n = 0$). (Alternately, we can beef up $\mathcal{L}(\text{PA})$ by throwing in a name $\bar{n}$ for every $n \in \mathbb{N}$. Sometimes this is preferable.) I won't bother with the $\bar{n}$ notation; in $\ulcorner\varphi(n)\urcorner$, I assume that n has been represented as a numeral before finding the code.

In this section and the next, I'll wave at some landmarks on the route to a real proof of Gödel's incompleteness theorem. We'll look both at PA and ZF—both are important. (Also, most everything in this section works if you want to discuss some other theory $\mathcal{T}$ formally inside either PA or ZF.)

The Coding System

Defining this is tedious but basically obvious. You have a countable number of symbols to deal with. So you assign a number (or a set) to each one. Lots of people use ordered pairs, say $(0, n)$ for the basic symbols $(,), \neg, \wedge, \exists, \ldots$, $(1, n)$ for the variables, $(2, n)$ for relation symbols, $(3, n)$ for function symbols, and $(4, n)$ for the name for n (if we have these). In PA you'd then use a computable pairing function to turn the pairs into numbers. In ZF you have the standard mechanism for pairs.

Computable (and Definable) Syntax

Formulas are finite sequences of symbols. Proofs are finite sequences of formulas. In PA, you need to able to do things like this:

- Determine which symbol a code stands for: (,), $\neg$, a variable, etc.
- Code finite sequences of numbers as single numbers. With this, you

have codes for terms, formulas, and proofs.

- Extract the length of a finite sequence, and its entries, from the code for the sequence.
- Determine whether a finite sequence of symbols is a term or a formula (or neither), and determine syntactic properties of coded terms and formulas, such as:
  - Is it closed? If not, what are its free variables?
  - Is it a conjunction $\varphi \wedge \psi$, and if so, what are the codes for φ and for ψ?
- Given a coded term $\ulcorner t \urcorner$ and coded formula $\ulcorner \varphi(x) \urcorner$ with a free variable, determine the code $\ulcorner \varphi(t) \urcorner$.
- Determine syntactic properties of finite sequences of formulas, such as whether it's a proof in PA, or more generally in the formal theory under consideration.

This isn't a complete list, but you get the idea.

It's intuitively clear that all the operations are computable, i.e., recursive. So invoking the results of §8.3, they are all definable in PA.

Turning to ZF, everything becomes so much easier. We no longer have to call on the Chinese remainder theorem to fix our problems, we have direct access to sequences. Some new issues do pop up when we turn from syntax to semantics; I'll discuss these in §10.1.

Now for some official handwaving. "It follows" from everything we've said that "$\ulcorner \pi \urcorner$ is a proof of $\ulcorner \varphi \urcorner$" and "PA $\vdash \ulcorner \varphi \urcorner$" can be expressed by formulas in PA (likewise for ZF). Textbooks fill out the details, mainly for the definability of all recursive functions. I've said all I'm going to on this topic.

9.2 Self Reference

Let's see how far we've advanced towards formalizing the Gödel sentence "I am unprovable."

$$\neg(\mathrm{PA} \vdash \ulcorner \mathrm{me} \urcorner)$$

Everything except "me" should seem unproblematic by now.

Gödel came up with another trick, related to the Cantor diagonal method. It's really a technique ("a trick you can use twice"). Suppose $\textsc{green}(x)$ is a formula of $\mathcal{L}(\mathrm{PA})$; so $\textsc{green}(\ulcorner \varphi \urcorner)$ expresses some property of φ, like unprovability—I'll just say "green". Now consider the function

$$\mathrm{sub}(n, \ulcorner \varphi(y) \urcorner) = \ulcorner \varphi(n) \urcorner$$

Wave your hands a bit—don't you believe that sub is computable? So it can be defined formally in $\mathcal{L}(\mathrm{PA})$, and

$$\textsc{green}(\textsc{sub}(x, x)) \tag{3}$$

says that the formula you obtain by plugging the numeral for x into the formula coded by x is green. Say n is the code for (3):

$$n = \ulcorner \textsc{green}(\textsc{sub}(x, x)) \urcorner$$

and look at

$$\ulcorner \textsc{green}(\textsc{sub}(n, n)) \urcorner \tag{4}$$

I claim that (4) is $\mathrm{sub}(n, n)$. Take a moment to see that this is true. (a) The formula coded by n is $\textsc{green}(\textsc{sub}(x, x))$. (b) We plugged the numeral for n into that formula.

So (4) says that (4) is green, i.e.,

> I am green.

This technique is really a diagonal argument. To see this more clearly, let $\psi_m(\vec{x})$ be the formula with Gödel number m: $\ulcorner\psi_m(\vec{x})\urcorner = m$. If ψ_m happens to have exactly one free variable, we have

$$\text{sub}(n, m) = \text{sub}(n, \ulcorner\psi_m(x)\urcorner) = \ulcorner\psi_m(n)\urcorner$$

and

$$\textsc{green}(\textsc{sub}(n, n)) \equiv \textsc{green}(\ulcorner\psi_n(n)\urcorner)$$

so if $\psi_n(x)$ is the formula $\textsc{green}(\ulcorner\psi_x(x)\urcorner)$, then $\psi_n(n)$ says "I am green".

Letting "green" be "unprovable" we have Gödel's original result: if the Gödel sentence (call it γ) is true, it's unprovable, so PA is **inadequate**—it fails to prove some true statements. If γ is provable, then it's false, so PA is **unsound**—it proves false statements.[6]

"Unsound" can be sharpened to "inconsistent". Thus: if PA proves γ, then the proof can be mirrored inside PA. That is, if π is a proof of γ, then

$$\text{PA} \vdash (\ulcorner\pi\urcorner \text{ proves } \ulcorner\gamma\urcorner)$$

and hence

$$\text{PA} \vdash (\exists x)x \text{ proves } \ulcorner\gamma\urcorner$$

But γ is equivalent to

$$\neg(\exists x)x \text{ proves } \ulcorner\gamma\urcorner$$

and so we have PA proving a statement and its negation. Summary: if PA proves γ, then PA is inconsistent.

Can "inadequate" be sharpened to "incomplete"? Not with the Gödel sentence. It is conceivable that PA proves $\neg\gamma$.[7] Let's assume this and see

[6] As a minor variation, let "green" be "my negation is provable". This is the version used in Stillwell [13, Ch.3].

[7] I chose "conceivable" with care. Nearly everyone believes that PA is sound, *a fortiori* consistent. Granted this, γ is both true and unprovable (from consistency) and $\neg\gamma$ is also unprovable (from soundness).

where it leads. If PA is consistent, then there *doesn't* exist any such proof of γ. So PA would be unsound, proving the false statement that there *is* a proof of γ. We've sharpened "inadequate": PA must be either inconsistent, incomplete, or unsound. (Inconsistent of course implies unsound, but not conversely.) In Gödel's original paper, he further sharpened "unsound" to a notion "ω-inconsistent". We needn't linger over that, because five years later Rosser came up with a variant of the Gödel sentence:

For any proof of me, there is a shorter proof of my negation.

or more formally (but omitting corner brackets for readabity):

$$(\forall\pi)[\pi \text{ proves me} \rightarrow (\exists\pi')(\text{length}(\pi') < \text{length}(\pi) \wedge \pi' \text{ proves } \neg\text{me})]$$

Let ρ be the Rosser sentence. Suppose that ρ or $\neg\rho$ is provable. Churn out all the proofs of PA in order of increasing length. If a proof of ρ pops up before one of $\neg\rho$, then ρ is false. Not only false, but this falsity can be proved in PA, since it's a "finitely verifiable" fact about manipulating symbols. So we would have a proof of $\neg\rho$ too. On the other hand, suppose a proof of $\neg\rho$ pops up first. If there's a proof of ρ of equal length, then we have proofs of both ρ and $\neg\rho$. If not, then we see that ρ is true—any eventual proof of ρ will be longer than the proof of $\neg\rho$ we just found—and again this truth is provable in PA. So in all cases if we have a proof of ρ or of $\neg\rho$, then we have a proof of the other and PA is inconsistent. Conclusion: PA is either inconsistent or incomplete.[8]

We complete our discussion of the Gödel's (first) incompleteness theorem with a grace note, known as his second incompleteness theorem. Con PA says that PA is consistent. How to mirror it inside PA? Gödel observed that

[8]Whereas Gödel had only shown that PA is either ω-inconsistent or incomplete. Another (more indirect) proof of Rosser's result had already been given using recursion theory: the set of theorems of a consistent complete recursively axiomatizable theory is recursive. If PA were complete and consistent, it would follow that the halting problem was decidable.

PA is consistent $\Leftrightarrow$ the Gödel sentence is true.

On the one hand, if PA is inconsistent, then *everything* is provable, including the Gödel sentence; but that means that the Gödel sentence is false. On the other hand, if if PA is consistent, then by the first incompleteness theorem, the Gödel sentence is unprovable and hence true. We could declare that the Gödel sentence itself is our formalization of Con PA! If this seems too clever by half, more natural formalizations of Con PA are provably equivalent to the Gödel sentence—the reasoning in this paragraph can be mirrored inside PA. Summary:

If PA is consistent, its consistency cannot be proven inside itself.

The story for ZF is almost word for word the same as for PA.

Gödel's self-reference technique has been chewed over many times. For my money, von Neumann's theory of self-reproducing machines sheds the most light. Imagine we have a warehouse filled with raw materials of all sorts in abundance (but no blueprints). We also have constructed a machine—a builder—that can take any blueprint we draw and use it to build the indicated machine. Imagine pasting the blueprint onto a blank plate in the builder and pressing the "go" button.

Our first attempt at a self-reproducing machine: we draw the blueprint for the builder, and press "go". It constructs another builder. But when we press the go button on the "daughter builder", it gives the error message "blueprint missing".

For our second attempt, we include a copy of the builder blueprint *in* the blueprint, much reduced in size, in the space for the plate. The daughter builder now ends up with a builder blueprint on its plate, and when the go button is pressed, it builds a granddaughter builder—a builder with no blueprint, since the miniaturized copy of the blueprint just showed a blank plate instead of a twice-reduced copy of the blueprint.

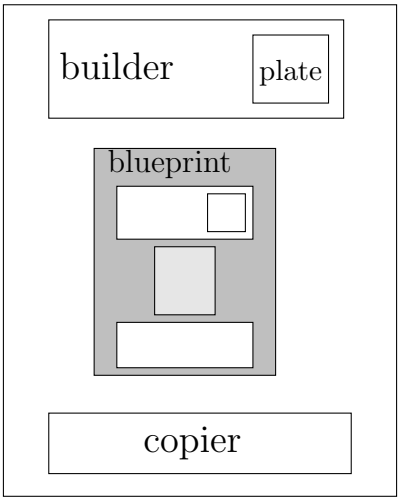

Figure 1: Self-Reproducing Machine

Are we looking at an infinite regress? No! We insert a (blueprint) copier *inside* the builder. We make a blueprint that shows the plate as blank, instead of containing a reduced copy. The blueprint also includes the blueprint for the copier. Then we put these instructions on the blueprint: "Build from the blueprint in the usual way, but when you're done, make a copy of your blueprint and paste it onto the blank plate of the constructed machine." That's our self-reproducing machine: a combination of a builder, a copier, and a blueprint with instructions. (See fig.1.)

The Gödel's sentence follows this model. The template $\mathrm{sub}(t, x)$ treats the x as an active blueprint: we use it to construct the formula $\varphi(y)$ such that $x = \ulcorner\varphi(y)\urcorner$. The t is a passive blueprint, just to be copied and pasted onto the "blank plate" y. Next, $\mathrm{sub}(x, x)$ says to use the same blueprint for both active and passive roles. Then $n = \ulcorner\textsc{sub}(x, x)\urcorner$ is the blueprint for all this, and $\mathrm{sub}(n, n)$ goes through the whole act of self-replication.

What is the role of φ in this? It's an extra piece of baggage, carried along

for the ride. If you like, think of it as a plaque, pasted on the outside of the machine: "This machine constructed by von Neumann industries", or "This machine is green".

10 Definability of Truth

Definability of truth in PA is the initial topic; then we expand our focus to the definability (or not) of other things.

10.1 What is Truth?

As for the title of this section, I couldn't resist. We first give Tarski's theorem on the undefinability of truth, then contrast it with some definitions.

True arithmetic, TA, is the set of all closed formulas satisfied by the standard model $(\mathbb{N}, +, \cdot, 0, 1)$. So the set of codes of formulas in TA is a subset of $\mathbb{N}$. Tarski showed that this subset is not definable in $\mathcal{L}(\text{PA})$. In other words, there is no formula $\textsc{true}(x)$ in $\mathcal{L}(\text{PA})$ such that for all closed formulas φ,

$$\mathbb{N} \models \varphi \Leftrightarrow \mathbb{N} \models \textsc{true}(\ulcorner \varphi \urcorner)$$

This follows immediately from the Gödel self-reference technique of §9.2: if there were such a predicate, then letting "green" be TRUE, we'd have the lying sentence

> I am false.

Call this E (for Epimenides). Then $\mathbb{N} \models E$ iff not $\mathbb{N} \models E$, contradiction.

But! In §3.3 we gave Tarski's definition of truth, for closed formulas in the language $\mathcal{L}(\mathcal{T})$ of a theory $\mathcal{T}$. It's an inductive definition, so at first blush

you'd think you could formalize it inside PA. Let's try, and see how far we get.

Because of the induction, we need to consider $\varphi(\vec{a})$ for any $\vec{a} \subseteq \mathbb{N}$. (I write $\vec{a} \subseteq \mathbb{N}$ to mean $a_i \in \mathbb{N}$ for each a_i of $\vec{a}$.) We can add names to $\mathcal{L}(\mathrm{PA})$, i.e., work with $\mathcal{L}_{\mathbb{N}}(\mathrm{PA})$. Or we can use numerals $1 + \cdots + 1$. Either way works.

The induction will start off with the atomic formulas. For PA, these are just formulas of the form $s = t$ for closed terms s and t. The set of codes of true equations $s = t$ is obviously recursive and so expressible in $\mathcal{L}(\mathrm{PA})$.

Now we try to phrase the Tarski definition formally. Let ψ, α, and β all stand for closed formulas, and $\varphi(x)$ for a formula with one free variable.

$$\begin{aligned}
&\textsc{true}(\ulcorner\psi\urcorner) \leftrightarrow \\
&\quad \ulcorner\psi\urcorner = \ulcorner s = t\urcorner \wedge \ s \text{ and } t \text{ are closed terms with equal values} \\
&\quad \vee\ \ulcorner\psi\urcorner = \ulcorner\neg\alpha\urcorner\ \wedge \neg\textsc{true}(\ulcorner\alpha\urcorner) \\
&\quad \vee\ \ulcorner\psi\urcorner = \ulcorner\alpha \wedge \beta\urcorner\ \wedge \textsc{true}(\ulcorner\alpha\urcorner) \wedge \textsc{true}(\ulcorner\beta\urcorner) \\
&\quad \vee\ \ulcorner\psi\urcorner = \ulcorner(\exists x)\varphi(x)\urcorner\ \wedge (\exists n)\textsc{true}(\ulcorner\varphi(n)\urcorner)
\end{aligned} \tag{5}$$

Most of the right hand side should not raise eyebrows. In the last clause, "$(\exists n)$" means "there exists a code for a numeral (or name) n". Plugging the numeral/name n into $\varphi(x)$ and finding the resulting code is a recursive procedure, as was noted earlier.

The one dubious aspect is the inductive definition itself. Does it fit into the $F(n+1) = f(F(n))$ paradigm from §8.3? Not obviously, and Tarski's undefinability theorem shows that the answer is no.

First solution: instead of a single predicate TRUE, we define a sequence of them $\textsc{true}_d$, where d stands for *depth* (or *degree*). That is, $\textsc{true}_d(\ulcorner\psi\urcorner)$ should be equivalent to $\mathbb{N} \models \psi$ only for closed formulas of degree $\leq d$ (or of parsing depth $\leq d$). So $\textsc{true}_0$ is truth for atomic formulas, $s = t$. We

rewrite the inductive definition as:

$$\begin{aligned}
&\textsc{true}_{d+1}(\ulcorner\psi\urcorner) \leftrightarrow \\
&\qquad \ulcorner\psi\urcorner = \ulcorner s = t\urcorner \wedge\ \textsc{true}_0(\ulcorner s = t\urcorner) \\
&\qquad \vee\ \ulcorner\psi\urcorner = \ulcorner\neg\alpha\urcorner\ \wedge \neg\textsc{true}_d(\ulcorner\alpha\urcorner) \\
&\qquad \vee\ \ulcorner\psi\urcorner = \ulcorner\alpha \wedge \beta\urcorner\ \wedge \textsc{true}_d(\ulcorner\alpha\urcorner) \wedge \textsc{true}_d(\ulcorner\beta\urcorner) \\
&\qquad \vee\ \ulcorner\psi\urcorner = \ulcorner(\exists x)\varphi(x)\urcorner\ \wedge (\exists n)\textsc{true}_d(\ulcorner\varphi(n)\urcorner)
\end{aligned} \tag{6}$$

This defines a sequence of formulas $\textsc{true}_d(x)$; the *induction* is outside PA, although the *formulas* belong to PA.

Second solution: we add a new predicate letter to $\mathcal{L}(\mathrm{PA})$, say T; let's call the augmented language $\mathcal{L}(\mathrm{PA}+T)$. A structure for this language consists of a structure for PA plus a subset of the universe. Here we consider the special case where the PA part is just the standard model, and we've picked some $A \subseteq \mathbb{N}$ to interpret T. Let's write $(\mathbb{N}, A)$ for this structure (leaving out $+$, etc., to reduce clutter).

Replace $\textsc{true}$ with T everywhere in (5). Then slap a universal quantifier $(\forall\ulcorner\psi\urcorner)$ around the whole thing. There's only one way that $(\mathbb{N}, A)$ can satisfy this formula, and that's to let A be the set of true formulas (i.e., codes of closed formulas true in the standard model).

Summarizing: there is no pure formula $\textsc{true}(x)$ defining truth for sentences in $\mathcal{L}(\mathrm{PA})$. But for any $d \in \mathbb{N}$, there is a pure formula $\textsc{true}_d(x)$ defining truth for sentences in $\mathcal{L}(\mathrm{PA})$ of parsing depth at most d. Finally, there is a pure closed formula $\textsc{truth}(T)$ in $\mathcal{L}(\mathrm{PA}+T)$ that defines the set T of true sentences of $\mathcal{L}(\mathrm{PA})$ implicitly: that is, $(\mathbb{N}, A) \models \textsc{truth}(T)$ iff A is the set of Gödel numbers of true sentences of $\mathcal{L}(\mathrm{PA})$.

10.2 Arithmetic and Hyperarithmetic Sets

Some terminology: suppose $A \subseteq \mathbb{N}$, and we have a formula $\varphi(x)$ in $\mathcal{L}(\mathrm{PA})$ such that

$$(\forall n)(n \in A \Leftrightarrow \mathbb{N} \models \varphi(n))$$

As we said in §8, such sets are called **arithmetic**. Tarski's undefinability theorem says that truth in $\mathbb{N}$ is not arithmetic.

As in the previous section, we consider $\mathcal{L}(\mathrm{PA})$ augmented with a new predicate letter, say S; we write $\mathcal{L}(\mathrm{PA} + S)$ for this language. Say we have a class $\mathcal{A}$ of subsets of $\mathbb{N}$, and a closed formula ψ_S in $\mathcal{L}(\mathrm{PA} + S)$. Suppose that

$$(\forall A \subseteq \mathbb{N})(A \in \mathcal{A} \Leftrightarrow (\mathbb{N}, A) \models \psi_S)$$

Then we say that $\mathcal{A}$ is an **arithmetic class**. If $\{A\}$ is an arithmetic class, then we say that A is **implicitly definable** by the formula ψ_S defining its singleton class. So truth in $\mathbb{N}$ is implicitly definable.

Observe that any arithmetic set is implicitly definable: just let ψ_S be $(\forall x)[S(x) \leftrightarrow \varphi(x)]$, where $\varphi(x)$ is the formula in $\mathcal{L}(\mathrm{PA})$ defining the arithmetic set.

The class of arithmetic sets enjoys some nice properties: it is closed under intersection, union, and complement. This falls out just from using the logical connectives. The existential quantifier gives us another operation, projection. Let $\langle y, z\rangle$ be a recursive pairing function, as in §8.2. If A is defined by $\varphi(x)$, then the formula $(\exists y)\varphi(\langle y, z\rangle)$ defines the projection of the relation $\langle y, z\rangle \in A$ onto its second coordinate. (If this relation is a function, then the projection is its range. Likewise, if the relation is a partial function, we can project on the first coordinate to find its domain.)

The class of arithmetic sets is, in fact, precisely the smallest class of subsets of $\mathbb{N}$ closed under intersection, union, complement, projection, and containing the singletons $\{0\}$ and $\{1\}$ and the graphs of the addition and

multiplication operations. (Take a moment to see why this is true.)

As we will see in §11, the class of implicitly definable sets is less tidy; for one thing, it's not closed under intersection.

We recover some of the nice features by passing to a yet more inclusive class, the *hyperarithmetic* sets. First we expand our language to so-called second-order arithmetic: instead of adding a new predicate letter, we allow variables ranging over subsets of $\mathbb{N}$. Quantification is allowed. I will use capital letters for subsets and lowercase for numbers[9]. I'll write both $S(x)$ and $x \in S$ to mean the same thing; you can decide which is vernacular. I'll write $\mathcal{L}^2(\text{PA})$ for second-order arithmetic.

Suppose A is implicitly definable via a formula ψ_S in $\mathcal{L}(\text{PA} + S)$. Then A is definable in $\mathcal{L}^2(\text{PA})$ in two ways, existentially:

$$(\exists S)[\psi_S \wedge x \in S]$$

and universally:

$$(\forall S)[\psi_S \to x \in S]$$

since in both forms, the clause "ψ_S" guarantees that S is the set A. Note that each formula has a sole free number variable, x.

A formula in $\mathcal{L}^2(\text{PA})$ is called a Σ^1_1 formula if it has an existential set quantifier out in front, no universal set quantifiers, and as many number quantifiers as your heart desires. Likewise, a Π^1_1 has a universal set quantifier out front, no existential set quantifiers, and as many number quantifiers as needed. Thus:

$$\begin{aligned} \Sigma^1_1: &\quad (\exists S)\psi(S, x) \\ \Pi^1_1: &\quad (\forall S)\psi(S, x) \end{aligned}$$

[9]So this is a two-sorted first-order language. Alternately, add predicates SET and NUMBER, and treat $(\exists X)\ldots$ as vernacular for $(\exists x)(\text{SET}(x) \wedge \ldots)$, etc.

where $\psi(S, x)$ can contain number quantifiers in abundance, but no other set quantifiers.

A set is **hyperarithmetic** or Δ^1_1 if it possesses both a Σ^1_1 and a Π^1_1 definition. (That is, Σ^1_1 and a Π^1_1 formulas with one free number variable, but no other free variables.) So all implicitly definable sets are hyperarithmetic. Feferman [2] showed that the converse is false; see §11.

I should say a bit about the Σ^1_1/Π^1_1 notation. Σ^0_n means the same as Σ_n as described in §8.5, likewise Π^0_n. A Σ^1_n formula has n alternating blocks of *set* quantifiers out front, starting with an $\exists X$; Π^1_n starts with a $\forall X$ block. Following the set quantifiers we can have any formula in $\mathcal{L}^2$(PA) *without* set quantifiers; i.e., "number quantifiers don't count". If you use the recursive tupling functions, you can replace quantifier blocks with single quantifiers (as I did in my definition of Σ^1_1 and Π^1_1). The terms **arithmetical hierarchy** and **analytical hierarchy** refer to the Σ^0_n/Π^0_n and Σ^1_n/Π^1_n classifications respectively; Rogers [9, Chs.14–16, pp.301–445] gives a full exposition.

We will need some familiarity with the the "definability powers" of arithmetic and hyperarithmetic sets in §11. Without giving an exhaustive treatment, here are some of the highlights.

We encountered some of the machinery in §8: recursive pairing functions, iterated to give n-tupling functions; the Gödel β function that encodes finite sequences of arbitrary length; definition by primitive recursion; the μ-operator. Using all this, any recursive relation or function can be expressed in $\mathcal{L}$(PA), i.e., is arithmetic. But the machinery has uses beyond this.

An example: although we've defined *arithmetic*, *implicitly defined*, and *hyperarithmetic* only for subsets of $\mathbb{N}$, we can expand the definitions to n-ary relations and functions via the n-tupling functions. (Alternately, we could allow n free number variables in the defining formulas; these approaches are equivalent.) Rogers [9, p.382] points out technical advantages to basing the

development on function variables instead of set variables. I will use both freely in $\mathcal{L}^2(\mathrm{PA})$.

Another example, which will play a role in §11.3. Regard a subset $A \subseteq \mathbb{N}$ as an infinite bitstring, or equivalently, a function $A : \mathbb{N} \to \{0, 1\}$. Imagine specifying A by a sequence of ever longer finite initial segments. Some terminology: a **condition** is a finite bitstring, or equivalently, a function $p : \{0, \ldots, k\} \to \{0, 1\}$. We use p, q, r for conditions. We say q **extends** p (written $q \succeq p$ or $p \preceq q$) if p is an initial segment of q. We say A **satisfies** p ($A \models p$) if p is an initial segment of A regarded as an infinite bitstring. We denote the empty condition (null string) by $\emptyset$. So $\emptyset \preceq p$ for all p, and every set satisfies $\emptyset$.

Suppose we have a sequence $p_0 \preceq p_1 \preceq \ldots$, defined inductively, starting with an arbitrary condition $p_0 = c$. In other words, we have a function $f : \mathbb{N} \times C \to C$, where C is the set of conditions, and for all $n \in \mathbb{N}$, $p_{n+1} = f(n, p_n)$. Let $A = \bigcup p_n$; if the p_n's stop growing at some point (i.e., for some n_0, we have $p_n = p_{n_0}$ for all $n \geq n_0$), then we make the rest of A all 0's after that (i.e., $n \notin A$ for all n past the end of p_{n_0}). Question: if f is arithmetic, or implicitly defined, or hyperarithmetic, can we say the same for A?

First, how do we code conditions? Using binary notation doesn't quite do the trick: consider 0001 vs. 01, for example. Pairing a binary number with its length works: $\langle k, b \rangle$ where k is the length of the bitstring, and b is the bitstring in binary, padded out with leading 0's if necessary. (Here $\langle , \rangle$ is the pairing function.)

A sketch of the formalization:

$$\begin{aligned} x \in S \leftrightarrow (\exists \text{ sequence } p_0, \ldots, p_l)\big[& \\ & p_0 = c \wedge \\ & (\forall n < l)[p_{n+1} = f(n, p_n)] \wedge \\ & x \in p_l \,\big] \end{aligned}$$

Let's look at the individual pieces:

- ($\exists$ sequence $p_0, \ldots, p_l$): we make use of Gödel's β function, which allows us to replace a quantification over finite sequences with one over numbers. Also, all the p_n's are conditions; we have to make this explicit.
- $p_0 = c$: c will be a constant, incorporated into our formula.
- $p_{n+1} = f(n, p_n)$: see below.
- $x \in p_l$: i.e., bit x in p_l is 1. This translates into a statement about the size of a remainder: b has a 1 in position x iff $b = 2^{x+1} \cdot q + r$ with $2^x \leq r < 2^{x+1}$.

Now, how about $p_{n+1} = f(n, p_n)$, or in general, $w = f(u, v)$? If f is an arithmetic function, defined by a formula $\varphi(w, u, v)$, then we just transcribe φ into our definition $\psi_S(x)$ for A. So if f is arithmetic, then so is A. (Because of c, we have a definition of A for each initial condition c, with $A \models c$.)

Next, say f is implicitly defined, say by φ_F; F is a new function symbol added to $\mathcal{L}(\mathrm{PA})$. (So φ_F is a closed formula of $\mathcal{L}(\mathrm{PA} + F)$.) Transcribing φ_F into the sketch above gives us a "paired" implicit definition. That is, $\{(f, A)\}$ is definable by a closed formula in $\mathcal{L}(\mathrm{PA} + F + S)$, where we've added two new symbols. As you can imagine, it's easy to combine f and A into a single function (or set), but that's not the same as having an implicit definition for A by itself.

Finally, if f is hyperarithmetic, then so is A. We transcribe the Σ^1_1 or the Π^1_1 formula for f into the sketch, getting a formula $\psi(x)$ in $\mathcal{L}^2(\mathrm{PA})$. The function (or set) quantifier can be migrated to the front, using basic facts of logic. So we have a Σ^1_1 and a Π^1_1 formula defining A.

Summary: if f is arithmetic, so is A; if f is hyperarithmetic, so is A; but if f is implicitly defined, then the best we can do is a paired implicit definition of A with f.

10.3 Definability in ZF

We turn from PA to ZF. Suppose we have a structure $\mathbf{S}$ for a language $\mathcal{L}$. For starters, let's assume that the universe S of $\mathbf{S}$ is a set and not a proper class. Now, ZF's purpose in life is to offer a stage for the formalization of intuitive mathematical arguments (so-called naive set theory). So we better be able to formalize truth for $\mathbf{S}$.

Pondering the definition of $\models$ given in §3.3, two desiderata come into focus:

- We need to formalize the notion of a closed formula in $\mathcal{L}_S$, i.e., $\mathcal{L}$ augmented with names for all elements of S.
- We need to formalize the inductive definition of $\mathbf{S} \models \varphi(\vec{a})$.

If S is a set, then the set of symbols of $\mathcal{L}_S$ is also a set, and the set of closed formulas in $\mathcal{L}_S$ is a subset of the set of all finite sequences of symbols in $\mathcal{L}_S$. (An inductively defined subset.)

Finite sequences pose no obstacles. The issue boils down to inductive definitions. All the inductions we need fit comfortably inside this framework:

1. We have a function $f : \omega \times A \to A$ for some set A, i.e., $f(n, a) \in A$ for all natural numbers n and all $a \in A$.
2. We claim for any $a_0 \in A$, there is a unique function $g : \omega \to A$ such

that

$$g(0) = a_0$$
$$g(n+1) = f(n, g(n)) \text{ for all } n \in \omega$$

For example, the set of formulas in $\mathcal{L}_S$: we let a_0 be the set of all atomic formulas, and let $f(n, a)$ be

$$a \cup \{\text{formulas of the form } \ulcorner\neg\alpha\urcorner, \ulcorner\alpha \wedge \beta\urcorner, \ulcorner(\exists x)\varphi(x)\urcorner$$
$$\text{where } \ulcorner\alpha\urcorner, \ulcorner\beta\urcorner, \ulcorner\varphi(x)\urcorner \in a \text{ and } x \text{ is a variable symbol}\}$$

High and wide mounds of tedious details lurk in that displayed equation. An inductive definition of terms must come before any talk of atomic formulas. The language $\mathcal{L}$ itself must provide a formal definition for 'variable symbol'. Etc. But I hope it's clear that only boredom and better things to do block the path to a complete formalization of the syntax of $\mathcal{L}_S$ in ZF.

And not just the syntax: $\models$ as well. For the syntax, we need only the set S and the language $\mathcal{L}$; for $\mathbf{S} \models \ldots$, the rest of the structure $\mathbf{S}$. This will consist of three functions: one mapping the constants to elements of S, another mapping relation symbols to relations, and the third mapping function symbols to functions. All set-theoretically pretty tame: every function has sets for domain and range, no proper classes anywhere in sight.

OK, what about the inductive framework? Here are two formalizations of "g is the function inductively defined by f and a_0":

$$(\forall \text{ functions } h : \omega \to A)$$
$$[h(0) = a_0 \wedge (\forall n \in \omega)h(n+1) = f(n, h(n)) \to h = g]$$
$$(\exists \text{ function } h : \omega \to A)$$
$$[h(0) = a_0 \wedge (\forall n \in \omega)h(n+1) = f(n, h(n)) \wedge h = g]$$

These are both formulas with four free variables: g, f, a_0, and A. It is a theorem of ZF that they are equivalent (that is, given the assumptions about f, A, and a_0).

We can now define $\mathbf{S} \models \varphi$ as a single formula $(\exists n \in \omega)\ulcorner\varphi\urcorner \in \text{True}(n)$, where $\text{True} : \omega \to \ldots$ is an inductively defined function.

Now let's drop the hypothesis that S is a set. To specify the structure $\mathbf{S}$, we now need proper classes for the relations and functions of $\mathbf{S}$. To make life easier, let's suppose $\mathcal{L}$ is finite, so we can simply list things. For example, if $\mathcal{L}$ has two relations P and Q, we could provide two formulas in ZF for the interpretations of P and Q in $\mathbf{S}$.

A specific example: say $\mathcal{L}$ is the $\mathcal{L}(LO)$, the language of linear orderings, and let the proper class S be On, the class of ordinal numbers. The symbol '$<$' is interpreted in On by the ZF formula $x \in y$. Better said: $(\text{On}, \in)$ is a class structure for $\mathcal{L}(LO)$.

The "collection" of all formulas in $\mathcal{L}_S$ will be a proper class. Our inductive framework creaks a little, but holds up with some tweaks. We have a proper class of atomic formulas, and of course things don't get better as we ascend in complexity. But we can still write a ZF formula for "φ is a formula in $\mathcal{L}_S$". The key idea: any *single* $\ulcorner\varphi\urcorner$ belongs to $\mathcal{L}_{S'}$ for some S' a subset of S (not just a set, but even a finite set). So φ is a formula in $\mathcal{L}_S$ iff $(\exists S' \text{ subset of } S)\ulcorner\varphi\urcorner \in$ formulas of $\mathcal{L}_{S'}$. The existential quantification is OK: if $\sigma(x)$ is a ZF formula defining the class S, then write $(\exists s)(\forall x \in s)[\sigma(x) \wedge \ldots]$. The '$\ldots$' is just "$\varphi$ is a formula of $\mathcal{L}_s$", OK for a set s.

Now what happens when we try to use this trick for $\models$? We stub our toes on existential quantification. $\mathbf{S} \models (\exists x)\varphi(x)$ is defined as $\mathbf{S} \models \varphi(a)$ for some $a \in S$. To state this in ZF requires an explicit quantification over the whole class S. Every additional quantifier in φ seeps over into an additional quantifier in the ZF formalization of $\mathbf{S} \models \varphi$.

Summarizing: in ZF, we can write a formula $\text{True}_n(\ulcorner\varphi\urcorner)$ that expresses $\mathbf{S} \models \varphi$ for all φ of complexity less than or equal to n. The inductive definition of the True_n takes place *outside* of ZF. The successive True_n have higher and higher complexity. If S is a set, we can migrate the induction

inside ZF and have a single formula True($\ulcorner\varphi\urcorner$). Now, even if S is a proper class, the *syntax* of $\mathcal{L}_S$ can be formalized in ZF because a single formula only involves a finite set of names. This doesn't work for $\models$ since the *semantics* of a single formula can still involve all of S. Tarski's undefinability theorem shows that there is no way around this obstacle, at least for $(V, \in)$ as a model of ZF.

To conclude this section, I'll say a bit about the role of definability in Gödel's class L of all the constructible sets. (I mentioned L in §4.6.) Suppose x is a set. A *definable subset* of x is a set of the form

$$\{z \in x : \varphi(z, \vec{a})\}$$

where $\varphi(z, \vec{t})$ is a formula in $\mathcal{L}(ZF)$ with free variables as shown, and $\vec{a} \subseteq x$. (Strictly speaking, I should say *definable with parameters*.) Let's denote the set of all definable subsets of x by $\mathcal{F}(x)$. (One can prove in ZF that there *is* a such a set.) Now we do a transfinite induction over all ordinals:

$$\begin{aligned} L_0 &= \varnothing \\ L_{\alpha+1} &= \mathcal{F}(L_\alpha) \\ L_\lambda &= \bigcup_{\alpha<\lambda} L_\alpha, \quad \lambda \text{ a limit ordinal} \\ L &= \bigcup_{\alpha\in\Omega} L_\alpha, \quad \Omega \text{ the class of all ordinals} \end{aligned}$$

This mirrors a standard construction in the formal development of ZF: replace $\mathcal{F}$ with the power set $\mathcal{P}$, and you get the so-called *cumulative hierarchy* V_α.

It's possible to code a lot of information into the parameters, and coupled with transfinite induction this gives a lot of "definability power". I won't go into more detail, but definability lies at the heart of Gödel's notion of a constructible set.

11 Forcing

Cohen introduced forcing in 1963 to prove the independence of the continuum hypothesis (CH) from ZFC. The next few years witnessed simplifications, other applications, and other perspectives. Despite the simplifications, set theory remains one of the most complicated arenas in which to encounter forcing for the first time. In this section, I'll look at forcing in a simpler context: Peano arithmetic.

Both in ZF and in the cases treated here, forcing concerns definability. Gödel had shown that if all sets are constructible, then AC and GCH hold. So Cohen started off by trying to show the relative consistency of the assumption, there exists a non-constructible set. As noted in §10.3, constructibility is all about definability.

In forcing in set theory, the basic concepts occur entangled with the cumulative hierarchy, the level-by-level build-up of the universe of sets from the empty set (see the end of §10.3). This makes for a messy situation. In Peano arithmetic, things aren't nearly so complicated. In §10.2 we looked at several kinds of definability for PA:

- *Arithmetic* subsets of $\mathbb{N}$. As we saw, "truth in PA" (the set of codes of true sentences in $\mathcal{L}(\mathrm{PA})$) is not arithmetic.
- *Implicitly definable* subsets of $\mathbb{N}$. As we saw, all arithmetic sets are implicitly definable; also, "truth in PA" is implicitly definable.
- *Arithmetic classes.* Implicit definability is the special case where a singleton $\{A\}$ is an arithmetic class.
- *Hyperarithmetic* subsets of $\mathbb{N}$ (aka Δ^1_1 subsets). As we saw, all implicitly definable subsets are hyperarithmetic.

Using forcing, Feferman [2] showed the existence of a hyperarithmetic set

that is not implicitly definable, and Addison [1] showed that the class of arithmetic sets is not an arithmetic class. For these results, we have to contend only with three levels—numbers, subsets of $\mathbb{N}$, and classes of subsets of $\mathbb{N}$—and not the whole hierarchy of ZF. I'll say a *little* bit more about ZF forcing in §11.7.

11.1 Intuition and Basic Definitions

We start by giving a rough idea of Cohen's notion of a generic set (in the PA setting); Feferman's and Addison's sets are generic. A generic set is *patternless* or *random*[10]; it has no special properties. For example, the set of primes is not generic, nor is any set containing the set of primes—that's a discernable pattern, even though it doesn't determine the whole set. Likewise, if a set contains *no* primes then it's not generic. The "least specific" thing we can say about $G \subseteq \mathbb{N}$, vis-a-vis primes, is that it contains an infinite number and omits an infinite number. So this should be true for any "generic" set. Perhaps you can already see how a generic set will be "slippery", eluding various kinds of definability.

A generic $G \subseteq \mathbb{N}$ can't literally have *no* properties. For example, either $0 \in G$ or $0 \notin G$, and likewise for any other number n. A **condition** ("formula" definition) is a consistent conjunction of a finite number of statements, each either $n \in G$ or $n \notin G$ for some $n \in \mathbb{N}$. (For example, $0 \in G \wedge 17 \in G \wedge 221 \notin G$. "Consistent" just means we don't have both $n \in G$ and $n \notin G$ in the same condition.) Suppose G satisfies a condition p. Cohen's methods enable us to say exactly which statements about G should hold, given that G is "generic" and satisfies p. Cohen used the (initially vague) notion of "generic" to inspire the notion of "forcing": if p forces φ, then φ holds for all generic G satisfying p.

[10]But we will avoid the word "random" from now on, since there is a different notion of *algorithmically random*.

The motivation runs from genericity to forcing, but the formal treatment take the reverse route: first an inductive definition of forcing, then a definition of generic set in terms of forcing, and finally some lemmas justifying the initial intuition.

Now let's nail down a few details. First, a matter of terminology. Shortly after Cohen's forcing appeared, Feferman [2] introduced a variation he called **weak forcing**. The one-way implication mentioned above becomes an equivalence for weak forcing: p weakly forces φ if and only if φ holds for all generic G satisfying p. For this reason, most authors nowadays just say "forcing" for "weak forcing". Cohen's version is, naturally, called "strong forcing". The notation $\Vdash^*$ is used for strong forcing, $\Vdash$ for weak forcing. I will follow this convention. We'll define (weak) forcing in §11.4. (The strong/weak distinction is independent of the ZF/PA difference.) As it happens, the most natural treatment first defines strong forcing, proves theorems about it, and then defines weak forcing as a derived concept.

We let $\mathcal{L}(\mathrm{PA} + G)$ be the language of PA augmented with one new unary predicate G. We write $(\mathbb{N}, A)$ for the structure consisting of the standard model $(\mathbb{N}, 0, 1, +, \cdot)$ of PA, plus $A \subseteq \mathbb{N}$ interpreting G. We let p, q, r stand for conditions. For our present purposes, conditions as bitstrings (as in §10.2) works better than the "formula" notion. Recall also the notations $p \preceq q$, $q \succeq p$, $A \models p$, and $\emptyset$ from §10.2: q extends p, A satisfies p, and the empty condition (i.e., null string). (One difference between the bitstring and formula notions: if $A \models p_1, p_2$, then with the bitstring notion, either $p_1 \preceq p_2$ or $p_2 \preceq p_1$. With the formula notion, we are assured only that p_1 and p_2 have a common extension q satisfied by A. We will avoid using this special fact about the bitstring notion, to keep certain arguments more generally applicable.)

We'll write $p \Vdash^* \varphi$ to mean that p strongly forces φ; here φ is a closed formula in $\mathcal{L}(\mathrm{PA} + G)$. The definition of $p \Vdash^* \varphi$ is inductive, as I mentioned. A key aspect of the definition: $\exists$ is regarded as basic, with $\forall$ just an abbreviation. Likewise, $\vee$ is basic with $\wedge$ an abbreviation.

1. $p \Vdash^* s = t \Leftrightarrow \mathbb{N} \models s = t$, where s and t are closed terms in $\mathcal{L}(\mathrm{PA})$.
2. $p \Vdash^* G(n) \Leftrightarrow p(n) = 1$.
3. $p \Vdash^* \alpha \vee \beta \Leftrightarrow p \Vdash^* \alpha$ or $p \Vdash^* \beta$.
4. $p \Vdash^* \neg\varphi \Leftrightarrow$ there is no $q \succeq p$ with $q \Vdash^* \varphi$.
5. $p \Vdash^* (\exists x)\varphi(x) \Leftrightarrow$ for some $n \in \mathbb{N}$, $p \Vdash^* \varphi(n)$.

In (2) and (5), we are being a bit sloppy about the distinction between $n \in \mathbb{N}$ and the numeral representing it in $\mathcal{L}(\mathrm{PA})$.

Let's talk about clause (4), the definition of $p \Vdash^* \neg\varphi$. Cohen originally put everything into prenex normal form. Dana Scott later suggested clause (4), simplifying the development. What is the intuitive justification for (4)? We want to have $p \Vdash^* \varphi$ precisely when $(\mathbb{N}, A) \models \varphi$ for all "generic" $A \models p$. Obviously then we don't want to have $p \Vdash^* \neg\varphi$ and $q \Vdash^* \varphi$ for some $q \succeq p$. On the other hand, if no extension $q \succeq p$ strongly forces φ, this indicates we will *always* find generic sets satisfying an extension q and also satisfying $\neg\varphi$, no matter how q extends p. Imagine going through the elements of $\mathbb{N}$ in increasing order, choosing which to put into A. Suppose at some point that $A \models p$. After that, we will *never* be able to assure the truth of φ; only when we've gone through "all of $\mathbb{N}$" will we find out if φ is true. That suggests that φ is somehow "specific", that we have to choose "carefully" infinitely often if we want φ to end up true. For example, suppose φ says "G contains only finitely many primes". This is more specific than its negation, since if n_0 is the largest prime in A, then we have to vigilantly exclude *every single prime* greater than n_0. So all generic A's will contain infinitely many primes (and likewise omit infinitely many).

A minor point, worth noting: we cannot have $p \Vdash^* \varphi$ and $p \Vdash^* \neg\varphi$. (Proof: exercise.)

Now we are ready to define "generic" formally.

> $A \subseteq \mathbb{N}$ is **generic** iff for every closed formula φ in $\mathcal{L}(\mathrm{PA} + G)$, there is a condition p with $A \models p$ and either $p \Vdash^* \varphi$ or $p \Vdash^* \neg\varphi$.

Intuitive justification: either $(\mathbb{N}, A) \models \varphi$ or $(\mathbb{N}, A) \models \neg\varphi$. Either way, the truth of φ or its negation should depend on only a finite amount of information about A, namely a condition p (plus the fact that A is "generic"). So we should have either $p \Vdash^* \varphi$ or $p \Vdash^* \neg\varphi$.

This suggests a slight extension of the $\Vdash^*$ notation. Write $A \Vdash^* \varphi$ to mean that there exists a p with $A \models p$ and $p \Vdash^* \varphi$. With this notation, we have

> $A \subseteq \mathbb{N}$ is generic iff for every closed formula φ in $\mathcal{L}(\mathrm{PA} + G)$, either $A \Vdash^* \varphi$ or $A \Vdash^* \neg\varphi$.

11.2 The Forcing Lemmas

Here are the key facts about forcing, as we've defined it. As before, φ is a closed formula in $\mathcal{L}(\mathrm{PA} + G)$.

Extension Lemma: If $p \Vdash^* \varphi$ and $q \succeq p$, then $q \Vdash^* \varphi$.

Truth Lemma: If A is generic, then $A \Vdash^* \varphi$ iff $(\mathbb{N}, A) \models \varphi$.

Empty Condition Lemma: If φ belongs to $\mathcal{L}(\mathrm{PA})$ (no G's), then $\emptyset \Vdash^* \varphi$ iff $\mathbb{N} \models \varphi$.

Definability Lemma: The relation $p \Vdash^* \varphi$ is implicitly definable, and hence hyperarithmetic. (Of course, the pairs (p, φ) have to be suitably coded.) Also, forcing for φ of complexity depth $\leq d$ is arithmetic, for every d.

Existence Lemma: There exists a generic set satisfying any given condition p_0.

The proofs of the first three lemmas amount to turning the "complexity induction" crank, with an occasional wrinkle. Start with the Extension Lemma. Suppose $q \succeq p$. Atomic sentences come in two varieties, $s = t$ and $G(n)$, and clauses (1) and (2) of the definition of $\Vdash^*$ dispose of both cases immediately. For $\neg\varphi$, "no extension of p does foo" implies "no extension of q does foo" because all extensions of q are extensions of p. Clause (3) makes short work of the case $\alpha \vee \beta$, and clause (5) of the case $(\exists x)\varphi(x)$.

Clause (4) and the Extension Lemma make a nice couple. Clause (4) says that $p \Vdash^* \neg\varphi$ is equivalent to "*no* extension of p strongly forces φ", while the Extension Lemma says that $p \Vdash^* \varphi$ is equivalent to "*every* extension of p strongly forces φ". If some but not all extensions of p strongly force φ, then p strongly forces neither φ nor its negation. Contrast this with the definition of a generic set.

The Extension Lemma also tells us that we cannot have both $A \Vdash^* \varphi$ and $A \Vdash^* \neg\varphi$, for *any* subset A (generic or not). Proof: if not, then we have $p_1 \Vdash^* \varphi$ and $p_2 \Vdash^* \neg\varphi$ for conditions p_1 and p_2 both satisfied by A. But p_1 and p_2 have a common extension, say q, so we would have $q \Vdash^* \varphi$ and $q \Vdash^* \neg\varphi$. As we've seen, this can't happen.

Now combine the previous paragraph with the definition of generic, and we have the important fact:

> For a generic A, $A \Vdash^* \neg\varphi$ iff not $A \Vdash^* \varphi$.

This makes the proof of Truth Lemma entirely routine. For the negation case:

$$A \Vdash^* \neg\varphi \Leftrightarrow \text{not } A \Vdash^* \varphi \Leftrightarrow \text{not } (\mathbb{N}, A) \models \varphi \Leftrightarrow (\mathbb{N}, A) \models \neg\varphi$$

with the middle $\Leftrightarrow$ being the inductive hypothesis. The other cases are handled in the same manner.

Next, the Empty Condition Lemma. The only case not utterly trivial is negation. The key is an "all or nothing" property for φ in $\mathcal{L}(\mathrm{PA})$: either *all* conditions strongly force φ, or *no* conditions strongly force φ. Given this, the induction is routine. The proof of the all or nothing property is itself an induction. Only negation calls for attention. Observe that if *all* conditions strongly force φ, then *no* conditions strongly force $\neg\varphi$, and if *no* conditions strongly force φ, then *all* conditions strongly force $\neg\varphi$. So the all or nothing property for φ entails it for $\neg\varphi$.

The Definability Lemma: The clauses of the definition of $\Vdash^*$ lend themselves to an implicit definition of the class $\{\langle \ulcorner p \urcorner, \ulcorner \varphi \urcorner\rangle : p \Vdash^* \varphi\}$. Here $\langle , \rangle$ is a recursive pairing function and $\ulcorner p \urcorner$ is the code for the condition as described in §10.2 (length plus binary). It's completely analogous to the implicit definition for truth we gave in §10.1: we add a new predicate FORCES and write out the clauses in the augmented language. We can also write a definition FORCES_d in $\mathcal{L}(\mathrm{PA})$ of forcing for all closed φ in $\mathcal{L}(\mathrm{PA}) + G$ of depth (or degree) less than or equal to d, again just like the predicates TRUE_d from §10.1.

Finally, the Existence Lemma. Enumerate all the closed formulas of $\mathcal{L}(\mathrm{PA})+G$, say $\{\varphi_n : n = 1, 2, \ldots\}$. We define a sequence of conditions $p_0 \preceq p_1 \preceq p_2 \preceq \ldots$ so that for each $n > 0$, either $p_n \Vdash^* \varphi_n$ or $p_n \Vdash^* \neg\varphi_n$. This is easy: p_0 is given to us. For p_{n+1}, either $p_n \Vdash^* \neg\varphi_{n+1}$ or $p_n \nVdash^* \neg\varphi_{n+1}$; in the latter case, some extension $q \succeq p_n$ strongly forces φ_{n+1}. In the first case let $p_{n+1} = p_n$, in the second let p_{n+1} be the first extension of p_n that strongly forces φ_{n+1}. (For "first", we can use any definable linear ordering of the conditions; the easiest is just to order them by their codes.) Now let $A = \bigcup p_n$, a set satisfying all the conditions. (Fill out the bitstring for A with trailing 0's in case the p_n's have an upper bound in length—but this can't happen, see next paragraph.) Obviously A is generic and satisfies p_0.

If the p_n's did have an upper length bound, then our construction would give a generic finite set A. Say all elements of A are less than n_0, so $(\mathbb{N}, A) \models (\forall n)[G(n) \to n < n_0]$. By the Truth Lemma, some $p \Vdash^* [G(n) \to n < n_0]$.

Now extend p to a q with a 1 in a spot after n_0, i.e., $q(n_1) = 1$ with $n_1 > n_0$. By the Extension Lemma $q \Vdash^* [G(n) \to n < n_0]$. Now use the Existence Lemma with $p_0 = q$, constructing a generic set $B \models q$. By the Truth Lemma applied to B, $(\mathbb{N}, B) \models [G(n) \to n < n_0]$. So all elements of B are less than n_0, but n_1 is an element of B, contradiction.

This is a typical use of the lemmas. Exercise: adapt the argument to show that any generic set contains and omits an infinite number of primes.

11.3 The Feferman and Addison Theorems

Any generic set A cannot be implicitly defined. Proof: suppose φ in $\mathcal{L}(\mathrm{PA}+G)$ defines the class $\{A\}$, i.e., $(\mathbb{N}, B) \models \varphi$ iff $B = A$. Since $(\mathbb{N}, A) \models \varphi$, by the Truth Lemma $A \Vdash^* \varphi$, i.e., $p \Vdash^* \varphi$ for some p with $A \models p$. But there certainly exists a $B \neq A$ also satisfying p, since p makes assertions about only finitely many numbers. By the Truth Lemma again, we must have $(\mathbb{N}, B) \models \varphi$ and so φ does not define $\{A\}$.

To prove Feferman's theorem, we just have to perform the Existence Lemma hyperarithmetically. By the Definability Lemma, the forcing relation is hyperarithmetic, i.e., there is a Δ^1_1 formula $\textsc{forces}(x, y)$ in $\mathcal{L}^2(\mathrm{PA})$ such that $p \Vdash^* \varphi$ iff $\mathbb{N} \models \textsc{forces}(\ulcorner p \urcorner, \ulcorner \varphi \urcorner)$, for all closed formulas φ in $\mathcal{L}^2(\mathrm{PA}+G)$. The inductive construction of the generic set A proceeds by defining ever longer conditions $p_0, p_1, \ldots$, with p_0 equal to some arbitrary condition c. The definition of p_{n+1} from p_n looks like this:

if $p_n \Vdash^* \neg\varphi_{n+1}$
    then $p_{n+1} = p_n$, and
if $q \succeq p_n$, $q \Vdash^* \varphi_{n+1}$, and q is the first such condition
    then $p_{n+1} = q$

or in symbols (and omitting the usual $\ulcorner\ \urcorner$ for readability):

$$\begin{aligned}
&[\text{FORCES}(p_n, \neg\varphi_{n+1}) \to p_{n+1} = p_n] \wedge \\
&(\forall q)[q \succeq p_n \ \wedge\ \text{FORCES}(q, \varphi_{n+1}) \wedge \\
&\qquad (\forall r)\,(r < q \wedge r \succeq p_n \ \to\ \neg\text{FORCES}(r, \varphi_{n+1})) \\
&\quad \to p_{n+1} = q]
\end{aligned}$$

We conclude that the function $p_{n+1} = f(n, p_n)$ is hyperarithmetic. (The formula obviously defines a hyperarithmetic relation; it's a function because of the negation clause in the definition of forcing.) Now, this is exactly the situation we examined at the end of §10.2. We saw that $\bigcup p_n$, which is A, is hyperarithmetic.

So Feferman's theorem is proved: there is a hyperarithmetic set that is not implicitly definable. Feferman actually proved a stronger result: there are an infinite number of "arithmetically independent" sets. I won't go into details.

We can squeeze a little more out of the proof we gave. Not only is the generic set A hyperarithmetic, but it can be "mutually implicitly defined" with the forcing relation. If we augment $\mathcal{L}(\text{PA})$ to $\mathcal{L}(\text{PA}+F+G)$ by adding two new predicates, then the pair $(\Vdash^*, A)$ can be implicitly defined: there is a closed formula in $\mathcal{L}(\text{PA} + F + G)$ which holds when and only when F represents the forcing relation and G represents the generic set A. It's no sweat to code the pair $(\Vdash^*, A)$ into a single $B \subseteq \mathbb{N}$, say by coding $\Vdash^*$ in the odd positions and A in the even ones. (So $p \Vdash^* \varphi$ iff $2\langle\ulcorner p\urcorner, \ulcorner\varphi\urcorner\rangle + 1 \in B$, and $x \in A$ iff $2x \in B$.) Thus B is implicitly defined. But B intersected with the even numbers is basically A, and is not implicitly definable. This proves an earlier remark, about how implicit definability is not closed under intersection.

A variation on the theme: Addison's theorem, which says that the class of arithmetic sets (call it $\mathcal{A}$) is not an arithmetic class. If $\mathcal{A}$ *were* an arithmetic class, that would mean there was a closed formula ψ in $\mathcal{L}(\text{PA} + G)$ such

that

$$A \in \mathcal{A} \Leftrightarrow (\mathbb{N}, A) \models \psi$$

Now, ψ has a certain complexity depth, say d. Although no generic set can be arithmetic, we *can* construct a "d-generic" arithmetic set, where:

> $A \subseteq \mathbb{N}$ is d-generic iff for every closed formula φ in $\mathcal{L}(\mathrm{PA} + G)$ of complexity depth $\leq d$, either $A \Vdash^* \varphi$ or $A \Vdash^* \neg\varphi$.

We just mimic the original proof of the Existence Lemma, but looking only at formulas of complexity depth $\leq d$, and using the fact that $\textsc{forces}_d$ is arithmetic, as noted in §11.2.

The proof of the Truth Lemma adapts immediately to show that

> If A is d-generic, and φ has complexity depth $\leq d$, then $A \Vdash^* \varphi$ iff $(\mathbb{N}, A) \models \varphi$.

Assume that A is d-generic and arithmetic, where d is the complexity depth of ψ, the formula defining the class $\mathcal{A}$ of arithmetic sets. Since A is arithmetic, $(\mathbb{N}, A) \models \psi$. By the adapted Truth Lemma, $A \Vdash^* \psi$, i.e., for some p with $A \models p$, we have $p \Vdash^* \psi$. Now let $B \models p$ be generic (not just d-generic). By the original Truth Lemma, this implies that $(\mathbb{N}, B) \models \psi$ and hence B is arithmetic. But we know that no generic set can be implicitly defined, let alone arithmetic. QED

Addison proved a somewhat stronger theorem; see Rogers [9, Ex.16–73,p.452] for the details.

11.4 Forcing Generalized

Cohen's work had immediate progency. People generalized and analyzed the framework. We explore some of that here.

Inspecting the theory of the last few sections uncovers this list of ingredients:

- A partially ordered set of *conditions*. Nowadays this partial order is called a *notion of forcing*.
- A collection of *structures*. Each condition furnishes some information about a possible structure.
- A *language* for making statements about the structures.
- An inductive definition of *forcing*, $p \Vdash^* \varphi$, where p is a condition and φ is a statement.
- The definition of a *generic* structure, and the *Truth Lemma* for generic structures: $G \models \varphi$ if and only if for some condition p, $G \models p$ and $p \Vdash^* \varphi$.
- The *Existence Lemma*, which constructs a generic structure satisfying a given condition.

Starting with the first bullet, let $(C, \preceq)$ be a partial order. We'll call the elements of C *conditions* and read $p \preceq q$ as q *extends* p. (This is the so-called Jerusalem convention. Many authors follow the so-called American convention, reversing the direction of the partial order.)

Next we generalize the set $A \subseteq \mathbb{N}$ of §§11.1–11.3: we replace A with the set of all conditions that it satisfies. (The notion $A \models p$ turns into $A \ni p$.) This set has two obvious properties: it is "closed downwards" ($A \ni q \succeq p$ implies $A \ni p$) and "directed upwards" (if $p, q \in A$, then there is an $r \in A$ with $p, q \preceq r$). Hence the following definition:

> A **filter** is a set of conditions that is closed downwards and directed upwards. An **ultrafilter** is a maximal filter, i.e., one not properly contained in any other filter.

An ultrafilter is our generalization of the notion of an infinite bitstring, i.e., a subset of $\mathbb{N}$. (Note that the definition works just fine with either the bitstring or the formula notion of condition.)

For the second bullet, I have nothing more specific to say; for the third, only that we have a first-order language. I'll write $A \models \varphi$ to indicate that "the structure built from A satisfies the formula φ", without trying to nail down the meaning of this. Of course, in any application the details must be spelled out. Your intuition is all you'll need here; for example, $A \models \neg\varphi$ iff not $A \models \varphi$.

For the fourth bullet, we copy the inductive definition of forcing from §11.1—except for clauses (1) and (2), which specify forcing for atomic sentences. *That* will depend on the specific application. (In ZF, it's a bit of a tangle.)

How should we define "generic filter"? Our earlier definition read:

> A is generic iff for every closed φ, there is a $p \in A$ such that either $p \Vdash^* \varphi$ or $p \Vdash^* \neg\varphi$.

(Replacing $A \models p$ with $p \in A$.) This will work, but let's analyze the requirement on p a little further. Define D_φ by:

$$D_\varphi = \{p \in C : p \Vdash^* \varphi \text{ or } p \Vdash^* \neg\varphi\}$$

In the proofs of the key lemmas of §11.2, we used this fact repeatedly:

$$(\forall p)(\exists q \succeq p) q \in D_\varphi$$

For if p does not strongly force $\neg\varphi$, then it has an extension q that strongly forces φ (see clause (4) of the definition of forcing). This motivates our next definition:

> A set D of conditions is **dense** iff $(\forall p)(\exists q \succeq p) q \in D$.

Suppose $\mathcal{D}$ is a collection of dense sets of conditions. Define:

> A is **generic for** $\mathcal{D}$ iff A intersects every set in $\mathcal{D}$, i.e., $A \cap D \neq \varnothing$ for all $D \in \mathcal{D}$.

We see that this generalizes our previous definition (with $\mathcal{D} = \{D_\varphi : \varphi \text{ is a closed formula}\}$).

All reference to the language has been sequestered into the definition of $\mathcal{D}$. The proof of the existence lemma turns into this:

> **Existence Lemma:** If $\mathcal{D}$ is a countable collection of dense sets of conditions, then for any condition p, there is a generic filter $A \ni p$.
>
> **Proof:** Let $\{D_n : n \geq 1\}$ be an enumeration of $\mathcal{D}$. Define a sequence $\{p_n : n \geq 0\}$ of conditions inductively by $p_0 = p$, and p_n is a condition extending p_{n-1} and belonging to D_n; p_n exists because D_n is dense. Let $A = \{q \in C : (\exists n) q \preceq p_n\}$. It is obvious that A is closed downwards and generic. If q, q' belong to A, then $q \preceq p_m$ and $q' \preceq p_n$ for some p_m, p_n, say with $m \leq n$. Then $q, q' \preceq p_n$, so A is directed upwards.

How about the Extension and Truth Lemmas? Assume A is a generic filter. Let's adopt the notation from the end of §11.1: $A \Vdash^* \varphi$ iff $p \Vdash^* \varphi$ for some $p \in A$. If you review the treatment in §11.2, you'll see we relied on the following:

- These lemmas can be verified for atomic sentences.
- A is directed upwards. This holds since A is a filter.
- $A \Vdash^* \neg\varphi$ iff not $A \Vdash^* \varphi$. If A is generic for $\mathcal{D}$, then this follows (as we've seen) if $\mathcal{D}$ includes all the D_φ.

So this framework includes a large part of our previous development. Our vagueness about the "generic structures" prevents us from saying anything more about atomic sentences, or generalizing the Empty Condition and Definability Lemmas.

11.5 Weak Forcing

The Truth Lemma says that $A \models \varphi$ iff $p \Vdash^* \varphi$ for some $p \in A$. How about a sort of obverse: is it true that $p \Vdash^* \varphi$ iff $A \models \varphi$ for all generic $A \ni p$? In one direction, yes:

> **Lemma:** If $p \Vdash^* \varphi$, then $A \models \varphi$ for all generic $A \ni p$.
>
> **Proof:** Suppose on the contrary that $p \Vdash^* \varphi$ but for some $A \ni p$, $A \models \neg\varphi$. By the Truth Lemma, $q \Vdash^* \neg\varphi$ for some $q \in A$. Since A is a filter, there is an $r \succeq p, q$, and by the Extension Lemma, $r \Vdash^* \varphi$ and $r \Vdash^* \neg\varphi$. But this is impossible by the way forcing is defined for negations.

But the other direction fails. The forcing of the previous sections provides a simple counterexample: let $\varphi \equiv (\exists x)G(x)$, and let p be the empty condition $\emptyset$. Then $\emptyset \Vdash^* (\exists x)G(x)$ iff $\emptyset \Vdash^* G(n)$ for some $n \in \mathbb{N}$ (the $\exists$ clause of the definition of forcing). That would mean *all* generic sets had to include this specific n, which isn't true. (Or consult the definition of $p \Vdash^* G(n)$: n must belong to the domain of the bitstring p with a 1 bit.) However, we saw that all generic sets are nonempty.

However, we do have a weaker equivalence:

> **Lemma:** $p \Vdash^* \neg\varphi$ if and only if $A \models \neg\varphi$ for all generic $A \ni p$.
>
> **Proof:** In one direction this is a special case of the previous lemma. For the other direction, we prove the contrapositive.

> Suppose not $p \Vdash^* \neg\varphi$. Then there is a $q \succeq p$ with $q \Vdash^* \varphi$. By the Existence Lemma, there is a generic filter $A \ni q$, and by the Truth Lemma, $A \models \varphi$. Since filters are closed downwards, $p \in A$, so not ($A \models \neg\varphi$ for all generic $A \ni p$).

A special case: $p \Vdash^* \neg\neg\varphi$ iff $A \models \varphi$ for all $A \ni p$. This motivates the definition:

> We say p **weakly forces** (or just **forces**) φ, written $p \Vdash \varphi$, if $p \Vdash^* \neg\neg\varphi$.

For weak forcing, we have two equivalences:

$$(a) : G \models \varphi \Leftrightarrow (\exists p \in G) p \Vdash \varphi$$
$$(b) : p \Vdash \varphi \Leftrightarrow (\forall G \ni p) G \models \varphi$$

This has such pleasing symmetry, that nowadays most authors just say "forces" instead of "weakly forces"; we follow suit. I will call these the **Fundamental Properties of Forcing**, and (b) the **Fundamental Property of Weak Forcing**. (We've already dubbed (a) the Truth Lemma.) Some authors use (b) as the definition of forcing; then they have to prove equivalence to an inductive definition, in order to prove the Definability Lemma.

Note that for negated statements, weak and strong forcing are equivalent: $p \Vdash \neg\varphi$ iff $p \Vdash^* \neg\varphi$. In other words, $p \Vdash^* \neg\neg\neg\varphi$ iff $p \Vdash^* \neg\varphi$. Another advantage of weak over strong forcing: $p \Vdash \neg\neg\varphi$ iff $p \Vdash \varphi$.

We unwind what $p \Vdash^* \neg\neg\varphi$ means:

$$\begin{aligned} & p \Vdash^* \neg\neg\varphi \\ \Leftrightarrow\ & (\forall q \succeq p) \neg\, q \Vdash^* \neg\varphi \\ \Leftrightarrow\ & (\forall q \succeq p) \neg (\forall r \succeq q) \neg\, r \Vdash^* \varphi \\ \Leftrightarrow\ & (\forall q \succeq p) (\exists r \succeq q) \neg\neg\, r \Vdash^* \varphi \\ \Leftrightarrow\ & (\forall q \succeq p) (\exists r \succeq q)\, r \Vdash^* \varphi \end{aligned}$$

This suggests another definition:

> D is **dense above** p iff $(\forall q \succeq p)(\exists r \succeq q) r \in D$. Thus, $p \Vdash \varphi$ iff the set of conditions that strongly force φ is dense above p.

Weak and strong forcing obey similar rules, with some key differences. Here's a (partial) list.

$$\begin{array}{lll} p \Vdash \neg\varphi & \Leftrightarrow (\forall q \succeq p) \text{ not } q \Vdash \varphi & (1) \\ p \Vdash \varphi \wedge \psi & \Leftrightarrow p \Vdash \varphi \text{ and } p \Vdash \psi & (2) \\ p \Vdash \varphi \text{ or } p \Vdash \psi & \Rightarrow p \Vdash \varphi \vee \psi & (3) \\ p \Vdash (\forall x)\varphi(x) & \Leftrightarrow (\forall a) p \Vdash \varphi(a) & (4) \\ p \Vdash (\exists x)\varphi(x) & \Leftrightarrow \{q \succeq p : (\exists a) q \Vdash \varphi(a)\} \text{ is dense above } p & (5) \end{array}$$

For contrast, we note:

$$\begin{array}{lll} p \Vdash \varphi \vee \psi & \not\Rightarrow p \Vdash \varphi \text{ or } p \Vdash \psi & (6) \\ p \Vdash^* (\forall x)\varphi(x) & \Leftrightarrow (\forall a)[\{q \succeq p : q \Vdash^* \varphi(a)\} \text{ is dense above } p] & (7) \\ p \Vdash^* \varphi \wedge \psi & \Leftrightarrow \{q \succeq p : q \Vdash^* \varphi \text{ and } q \Vdash^* \psi\} \text{ is dense above } p & (8) \end{array}$$

The demonstrations flow mostly from the Fundamental Properties of Forcing, (a) and (b) above.

1. Plug $\neg\neg\varphi$ in for φ in the negation rule for $\Vdash^*$.

2. Use (a). Alternately, (b) and the (easily shown) fact that the intersection of two open sets that are dense above p is dense above p. (Recall that "open" means "upwards closed".)

3. Use (a), or (b) and the fact that if $A \subseteq B$ and A is dense above p, then B is also dense above p.

4. Use (a), interchanging the universal quantifiers $(\forall G \ni p)$ and $(\forall a)$.

5. Using (b), it's enough to show that for the two sets

$$\{q \succeq p : (\exists a) q \Vdash^* \varphi(a)\}$$
$$\{q \succeq p : (\exists a) q \Vdash \varphi(a)\}$$

one is dense above p iff the other is. Since strong forcing implies weak forcing, one direction is obvious. For the other direction, suppose that $\{q \succeq p : (\exists a) q \Vdash \varphi(a)\}$ is dense above p. Written out, using (b) on $q \Vdash \varphi(a)$:

$$(\forall r \succeq p)(\exists q \succeq r)(\exists a)(\forall s \succeq q)(\exists t \succeq s) t \Vdash^* \varphi(a)$$

Suppose that's true. For arbitrary $r \succeq p$, choose a $q \succeq r$ and an a such that $(\forall s \succeq q)(\exists t \succeq s) t \Vdash^* \varphi(a)$. In particular, we can choose $s = q$, so there is a $t \succeq q \succeq r$ with $t \Vdash^* \varphi(a)$. Which is to say,

$$(\forall r \succeq p)(\exists a)(\exists t \succeq r) t \Vdash^* \varphi(a)$$

i.e., $\{q \succeq p : (\exists a) q \Vdash^* \varphi(a)\}$ is dense above p.

6. For example, we have $\emptyset \Vdash G(0) \vee \neg G(0)$, since every set (let alone every generic set) satisfies $0 \in G \vee 0 \notin G$. But $\emptyset \nVdash G(0)$ and $\emptyset \nVdash \neg G(0)$, since neither holds for all generic sets.

   Since strong forcing implies weak forcing, we have $\emptyset \nVdash^* G(0)$ and $\emptyset \nVdash^* \neg G(0)$. But strong forcing distributes over disjunctions, so we have a corollary: $\emptyset \nVdash^* G(0) \vee \neg G(0)$. Another strong forcing oddity.

7. Since $(\forall x)\varphi(x)$ is an abbreviation for $\neg(\exists x)\neg\varphi(x)$, and strong and weak forcing are equivalent for negated statements, we have

$$p \Vdash^* (\forall x)\varphi(x) \Leftrightarrow p \Vdash (\forall x)\varphi(x) \Leftrightarrow (\forall a)p \Vdash \varphi(a)$$

   By (b), the last is equivalent to

$$(\forall a)\{q \succeq p : q \Vdash^* \varphi(a)\} \text{ is dense above } p$$

8. Basically the same argument as just given for universal quantifiers.

Summarizing, weak forcing looks more like classical logic. For forcing in PA, we can give a direct inductive definition of weak forcing by using (1), (2), and (4), plus the same rules for atomic formulas: $p \Vdash s = t$ iff $\mathbb{N} \models s = t$, and $p \Vdash G(n)$ iff $p(n) = 1$. (Regard $\forall$ and $\wedge$ as basic, with $\vee$ and $\exists$ as derived.)

Why then bother with strong forcing? First, it was Cohen's original version, so historical interest. Second, one of the nicer treatments of set theory forcing [10] uses it. Finally, it has close ties to intuitionistic logic, so it may be worth knowing for that.

11.6 Boolean Algebras

Any partial order satisfying a mild condition can be embedded as a dense subset of a complete boolean algebra; the condition typically holds for notions of forcing. This construction is fundamental for the boolean algebra approach to forcing. I could run through the definitions in a paragraph or two, but they would make little sense without a longer discussion. It's more helpful, I think, to outline a simple example, leaving a full treatment for the textbooks (e.g., [3, §7,§14], [5, §II.3]).

Let C be the set of all functions from finite subsets of $\mathbb{N}$ to $\{0, 1\}$; we call elements of C *conditions*, of course. Let X be the set of all functions from $\mathbb{N}$

to $\{0,1\}$; let's call these *bitstrings*. Every condition p determines a subset $[p]$ of X, namely all the bitstrings satifying it. The collection of all the $[p]$'s will serve as a base for a topology on X, because either $[p] \cap [q] = \varnothing$ or $[p] \cap [q] = [p \cup q]$ (with the obvious meaning for $p \cup q$). Let T be this topology.

Given any topology T on any space X, the so-called regular open sets form a boolean algebra. First we define an operation to play the role of boolean complement: $-U = X \setminus \overline{U}$, i.e., the complement of the closure. An open set U is *regular* if $-(-U) = U$. It turns out that if U is any open set, then $-(-U)$ is regular. Define $U \wedge V = U \cap V$ and $U \vee V = -(-(U \cup V))$. The collection of all regular open sets with the operations $\wedge$, $\vee$, and $-$ forms a boolean algebra.

If you think about for a moment, you'll realize that $-[p]$ is the union of all the $[q]$'s such that q is incompatible with p. It's not hard to see that each $[p]$ is regular, and so the map $p \mapsto [p]$ embeds C as a subset of our boolean algebra; it is *dense*, with an appropriate definition of the term (for boolean algebras).

C becomes a partial order if we define $p \preceq q$ when q extends p, i.e., $\mathrm{dom}(p) \subseteq \mathrm{dom}(q)$ and q agrees with p on p's domain. This adheres to the Jerusalem convention of §11.4. But the standard partial order for T is inclusion: $U \leq V$ iff $U \subseteq V$, giving us $U \wedge V \leq U, V$ and $U, V \leq U \vee V$. With these definitions, the map $p \mapsto [p]$ is an *order-reversing* embedding of C into the boolean algebra. For this reason, many authors follow the American convention and define $p \preceq q$ to mean that p extends q.

Stepping back from the details, we see that a condition provides a finite amount of information about a bitstring; an open set offers us a (possibly infinite) disjunction of conditions. The empty open set represents impossibility; that is, no bitstring satisfies it. By contrast, the empty condition corresponds to the entire space X. Using the American convention, the empty condition is the largest condition; C doesn't have a smallest condi-

tion, but if we added one, it would correspond to $\varnothing$.

What's the deal with regular open sets? Say U is an open set, and x belongs to U; so to speak, x satisfies U's disjunction. Therefore x satisfies some p with $[p] \subseteq U$. If U is regular open, then this also holds: if x does not belong to U, then x satisfies some p with $[p] \cap U = \varnothing$. So we have a "finite information" property both positively and negatively.

11.7 Other Formulations and Applications

This section sketches Robinson's forcing in model theory, and ZF forcing. (Really just silhouettes, with very little detail.)

Abraham Robinson applied Cohen's forcing to model theory. In fact, he defined two variants, called infinite forcing and finite forcing. Finite forcing turned out to be the suppler concept. Let $\mathcal{T}$ be a theory in a first-order language $\mathcal{L}$. Augment $\mathcal{L}$ to $\mathcal{L}_A$ by adding a countable set A of new constants we'll call *names*. We let $\mathcal{T}_A$ be $\mathcal{T}$ over $\mathcal{L}_A$, i.e., it has the same postulates. (What's the difference? A model of $\mathcal{T}_A$ must have assigned values for all the names.) A *basic* statement is a variable-free formula in $\mathcal{L}_A$ that is either atomic or the negation of an atomic statement. (Examples in $\mathcal{L}_A$(PA): $a_1 + a_2 = a_3$; $\neg(a_1 < 1 + 1)$.) A *condition* p is a consistent conjunction of a finite number of basic statements; "consistent" means that p holds in some model of $\mathcal{T}_A$.

Robinson mimicked Cohen's inductive definition of forcing, leading to the definition of a *finitely generic structure* for $\mathcal{T}_A$: $\mathbf{M}$ finitely generic means that for all closed formulas φ in $\mathcal{L}_A$, there is a condition p satisfied by $\mathbf{M}$ such that either $p \Vdash^* \varphi$ or $p \Vdash^* \neg\varphi$. Some of the forcing lemmas of §11.2 have analogues. but not all. As it happens, a finitely generic structure for $\mathcal{T}_A$ does *not* always satisfy $\mathcal{T}$. (If $\mathcal{T}$ is "nice enough"—technically, is what's called an inductive theory—then all finitely generic structures for $\mathcal{T}_A$ are models of $\mathcal{T}$.) The theory of the finitely generic structures is called the

finite forcing companion of $\mathcal{T}$.

Robinson manufactured the finitely generic structures by starting with the universe of all variable-free terms built from the constants of $\mathcal{L}_A$ (i.e., all built-in constants of $\mathcal{L}$, plus all the names). The construction of the Existence Lemma gives a so-called generic sequence of conditions, and this sequence determines all the relations: $R(\vec{t})$ holds iff it appears in the generic sequence. So Robinson forcing is only mildly more elaborate than what we've seen in §11.1. ZF is another story.

Cohen's original set up: let $\mathbf{M}$ be a countable standard model of ZF. That is, $\mathbf{M} = (M, \in)$ where M is a countable subset of the "true" universe V, and $\in$ is the standard element-of relation. The assumption that there *is* such a countable standard model is called Axiom SM; it follows informally from the downward Löwenheim-Skolem theorem, applied to $(V, \in)$, but this argument cannot be carried out formally inside ZF. The use of Axiom SM smooths the exposition; it can be avoided with a little bit of work.

Cohen wished to extend $\mathbf{M}$ to a model $\mathbf{M}[G]$ containing a new set $G \subseteq \omega$. (Because M is countable, we know there are subsets of ω not in M.) The goal was to make G not constructible "in $\mathbf{M}[G]$". Gödel had devised a formula $\Lambda(x)$ in $\mathcal{L}(\text{ZF})$ formalizing "x is constructible". So the idea is that $\mathbf{M}[G] \models \text{ZF}$ but $\mathbf{M}[G] \models \neg\Lambda(G)$.

Now Cohen ran up against a number of hurdles. First, you can't just add one set to $\mathbf{M}$ and get a model of ZF. The axioms of ZF allow you to build on the new set G in a dizzying variety of ways: the pair (G, G), the product $G \times G$, $G \cup X$ and $G \cap X$ for all the $X \in M$, things like $\bigcup_{n\in\omega} G^n$, all sorts of functions from or to G or anything you build from it, etc. And you can iterate all this transfinitely! It's a *little* like adjoining a new element to a field, but much messier.

When we adjoin a new element r to a field F, we first look at all the polynomials $a_0 + a_1 r + \cdots + a_n r^n$ with coefficients in F. Can we do something

similar here? We'd need a way of describing all possible sets that can be built out of G. Cohen's original approach adapted the constructible hierarchy, heaping all the technicalities of Gödel's work on top of his own new tricks. Scott and Solovay, and independently Vopěnka simplified matters by using boolean-valued models. Later Shoenfield reverse-engineered this approach, essentially replacing the constructible hierarchy L_α (or rather Cohen's adaptation of it) with the cumulative hierarchy V_α.

Whatever approach you take, you end up with "names" or "descriptions" for the elements of $\mathbf{M}[G]$, but these are way more complex than the names of Robinson forcing. This has a knock-on effect: defining forcing for atomic assertions ($p \Vdash^* a = b$ and $p \Vdash^* a \in b$, a and b being "names") requires a convoluted double transfinite induction, which rears its head when proving any of the forcing lemmas. Altogether, probably the most intricate setting in which to learn about forcing.

'Nuff said!

References

[1] J. W. Addison. The undefinability of the definable. *Notices of the American Mathematical Society*, page 347, 1965. Abstract 622-71.

[2] Solomon Feferman. Some applications of the notions of forcing and generic sets. *Fundamenta mathematicae*, 56(3):325–345, 1965.

[3] Thomas Jech. *Set Theory: The Third Millennium Edition*. Springer, 2002.

[4] Richard Kaye. *Models of Peano Arithmetic*, volume 15 of *Oxford Logic Guides*. Oxford University Press, 1991.

[5] Kenneth Kunen. *Set Theory: An Introduction to Independence Proofs*. Elsevier, 1980.

[6] Gregory Moore. The origins of Zermelo's axiomatization of set theory. *Journal of Philosophical Logic*, 7(1):307–329, January 1978.

[7] Gregory Moore. *Zermelo's Axiom of Choice: Its Origins, Development, and Influence*. Springer, 1982.

[8] W. V. Quine. Concatenation as a Basis for Arithmetic. *Journal of Symbolic Logic*, 11(4):105–114, 1946.

[9] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967.

[10] J. R. Shoenfield. Unramified Forcing. In Dana Scott, editor, *Axiomatic Set Theory*, volume 13 part 1, pages 357–381. American Mathematical Society, 1971.

[11] Raymond M. Smullyan. *Theory of Formal Systems*. Princeton University Press, 1961.

[12] Raymond M. Smullyan and Melvin Fitting. *Set Theory and the Continuum Problem*. Dover, 2010.

[13] John Stillwell. *Roads to Infinity*. A K Peters, 2010.