CDM

Reasoning and Proving

Klaus Sutner
Carnegie Mellon University
www.cs.cmu.edu/~sutner

Battleplan

- Entscheidungsproblem
- Reasoning Systems
- Weird Examples
- Formalizing Reasoning
- Frege's System
- Deduction Systems
- Automatic Theorem Provers

Entscheidungsproblem

Hilbert's Entscheidungsproblem

> The Entscheidungsproblem is solved when one knows a procedure by which one can decide in a finite number of operations whether a given logical expression is generally valid or is satisfiable. The solution of the Entscheidungsproblem is of fundamental importance for the theory of all fields, the theorems of which are at all capable of logical development from finitely many axioms.
>
> D. Hilbert, W. Ackermann
> Grundzüge der theoretischen Logik, 1928

Decision Problems

In modern terminology: Hilbert and Ackermann are trying to solve a decision problem: is such and such assertion valid?

The important point is that they want a *decision procedure*, an algorithm to answer the question.

The assertions could be taken from anywhere in mathematics (or at least arithmetic, group theory, topology, . . .).

Note, though, the slight hedging: the field has to be "capable of logical development" and the number of fundamental assumptions (axioms) must be finite.

As we will see, pinning down exactly what is meant by logical development is not as easy as one might think.

Polynomials and the Entscheidungsproblem

For example, suppose we are interested in the existence of a root of a multivariate polynomial with integral coefficients:

$$p(x_1, x_2, \ldots, x_n) = 0$$

How hard is to check if a root exists? It depends on what kind of roots we are interested in:

- For integer roots the problem is undecidable by a famous result of Y. Matiyasevich in 1970.
- For real roots the problem is decidable by another famous result of A. Tarski in 1951. Likewise for complex roots.
- For rational roots the problem is currently open.

Dealing with Undecidability

Undecidability of a general problem tends to cast a shadow even when one deals with special cases: the theory of Diophantine equations is very difficult.

For example, the seemingly innocent equation

$$x^2 - 991y^2 - 1 = 0.$$

has $x = 1$, $y = 0$ as a trivial integral solution. The least positive integral solution here is

$$\begin{aligned} x &= 379516400906811930638014896080 \\ y &= 12055735790331359447442538767 \end{aligned}$$

This solution is easy to verify (at least if one has access to an arbitrary precision arithmetic package), but not quite so easy to find.

Efficient Entscheidungsproblem

Even if there is an algorithm that solves the Entscheidungsproblem in a particular domain it may be of dubious practical value.

A classical example is Satisfiability, the problem of deciding whether a formula

$$\varphi(x_1, x_2, \ldots, x_n)$$

of propositional logic is satisfiable (i.e., whether there is a truth assignment $\mathrm{Var} \longrightarrow \mathbf{2}$ that models φ).

Satisfiability is trivially decidable: we can simply check all 2^n possible truth assignments. However, it seems that there is no algorithm that runs in time polynomial in n, the number of variables (if there were such an algorithm we would have $\mathbb{NP} = \mathbb{P}$, an unlikely proposition).

On the other hand, there are excellent algorithms (Davis/Putnam) that can handle formulae with thousands of variables and hundreds of thousands of clauses on a routine basis (at least if the formula is of natural origin).

Reasoning

In the absence of a (known) decision algorithm we may still be able to tackle individual cases of a problem by some kind of argument, by a reasoning process. Even if a general method is elusive we may succeed with many interesting results special cases (dealing with them on a strictly case by case basis).

For example, though elementary number theory is not decidable a wealth of information has been accumulated over the last 2500 years and many amazing facts are known. Perhaps the most recent addition to the list of outstanding results in number theory is A. Wiles' proof of *Fermat's Last Theorem*:

$$x^n + y^n = z^n$$

has no solution in positive integers when $n > 2$.

How can we make certain that this assertion is correct?

First a few more mundane examples.

Reasoning 1

Everyone knows that the law of associativity

$$x + (y + z) = (x + y) + z$$

implies that we can "place the parentheses arbitrarily" in an expression. But what exactly does this mean? How does the second assertion (once clarified) follow from the first?

Likewise, it is "clear" that commutativity

$$x + y = y + x$$

does not follow from associativity. But exactly why is commutativity not a consequence of associativity?

This particular problem is not too hard to tackle, but we would like a systematic method that works for more complicated problems.

Reasoning 2

Consider the following (toy-)specifications for an airplane.

- outside temp. $< -50^o\ C$ implies hydraulic system partially impaired
- hydraulic system partially impaired implies landing gear takes at least one minute to deploy
- landing gear must deploy in at most 30 seconds under all circumstances
- aircraft must be able to fly on all standard routes

Given the additional fact that, on some standard routes, outside temperatures drop below $-50^o\ C$, these specifications are self-contradictory.

This is easy to see, but what if the specification documents for an airplane wouldn't even fit inside the airplane? A good part of these specifications would be stated in much less concise and clear language, too.

Reasoning 3

Here is a classical program that computes the GCD of two positive integers:

```
while( y != 0 )
       ( x, y ) = ( y, x mod y );
return x;
```

To show that this little piece of code works as intended we need to establish

- Correctness: typically using loop invariant such as "the GCD of x and y remains unchanged"
- Termination: after finitely many executions of the loop body we have $y = 0$.

This is not too hard, but it requires work and some background information in number theory. What if we replace the integers by polynomials, a generalization that is very important in cryptography?

Reasoning 4

Array access can be somewhat complicated even in simple algorithms such as the Cocke-Younger-Kasami algorithm for context-free parsing (a dynamic programming method).

for $i = 1$ to n **do**
    $t_{i,i} = \{ A \mid A \to w_i \}$;

for $d = 1$ to $n - 1$ **do**
for $i = 1$ to $n - d$ **do**
    $j = d + i$;
    for $k = i$ to $j - 1$ **do**
        if exists $A \to BC$, $B \in t_{i,k}$, $C \in t_{k+1,j}$
        then add A to $t_{i,j}$;

How can one prove that the array elements are accessed in the right order? The main problem here is that one needs to keep track of many simple arithmetic operations and various constraints.

Ariane V

The Disaster

On June 4, 1996, the maiden flight of the European Ariane 5 rocket crashed about 40 seconds after takeoff. The direct financial loss was about half a billion dollars, all uninsured. An investigation showed that the explosion was the result of a software error.

Particularly annoying is the fact that the problem came from a piece of the software that was not needed during the crash, the Inertial Reference System. Before lift-off certain computations are performed to align the IRS. Normally they should be stopped at -9 seconds. However, a hold in the countdown would require a reset of the IRS taking several hours. So, the computation continues for 50 seconds after the start of flight mode. In the Ariane 5, which was larger than its predecessors, this caused an uncaught exception due to a floating-point error: a conversion from a 64-bit integer to a 16-bit signed integer, which should have been a number less than 2^{15}, was erroneously applied to a greater number (the "horizontal bias" of the flight).

There was no explicit exception handler (since it presumably was not needed), so the entire software crashed and the launcher with it.

Reasoning Systems

E. Schröder on Consequences

Historically, more than a century ago, some mathematicians recognized very clearly that some systematic understanding of the mechanism of inference is essential.

> Getting a handle on the consequences of any premises, or at least the fastest methods for obtaining these consequences, seem to me to be the noblest, if not the ultimate goal of mathematics and logic.
>
> E. Schröder, Algebra und Logik der Relative, 1895

Note the date. If Schröder were to write his book today, computer science would also appear on this list.

Also note the reference to "fastest methods". The intent here was to develop good methods for hand calculations; but efficiency is even more important for machine calculations (which are often applied to much larger instances).

Gottfried Wilhelm von Leibniz (1646–1716)

In fact, even three centuries ago Leibniz speculated about the possibility of a "calculus ratiocinator", a universal system that would render all human thought amenable to calculation.

> ". . . a kind of general system of notation in which all the truths of reason should be reduced to a calculus. This could be, at the same time, a kind of universal written language, very different from all those which have been projected hitherto; for the characters and even the words would direct the reason, and the errors – excepting those of fact – would only be errors of calculation. It would be very difficult to invent this language or characteristic, but very easy to learn it without any dictionaries."

It is no coincidence that Leibniz also invented a superior notation system for differential calculus.

Alas, he never developed anything resembling a full system (he was rather busy, being by turns a philosopher, scientist, mathematician, diplomat, librarian, and lawyer).

Automatic Reasoning

One should note that the design of a complex system such as the Ariane 5 could not be verified by hand, even if the specifications had all been available and expressed in absolute precision. In many "real world" applications one encounters structures that are exceedingly complicated, specifications comprising thousands of pages, and software programs millions of lines long. Handling these systems without computer assistance is entirely hopeless, no matter how elegant a calculus we try to apply (even if Leibniz had finished his calculus ratiocinator project).

So, we need not just a formal system, a deductive calculus (at least from the computer science perspective, pure mathematicians might disagree) but we want systems that lend themselves to implementation and that allow for fast algorithms.

This requirement adds yet another layer of difficulties; pure decidability result are not sufficient, we want efficient methods. Unsurprisingly, it is best to ignore these issues and initially focus and the clean theory, the headaches can be added later.

Truth and Proof

So, for the time being, let's ignore pesky applications and focus on easier examples in mathematics. Why would anyone care about a formal description of some mathematical fact?

Why not simply use the informal but rigorous language, together with sound intuition, that has served mathematics well for many centuries, even millennia? For example, Euclid's proof of the infinity of primes is perfectly valid some 2500 years after its discovery. It can be used in an argument very much like any theorem proven just a few years ago. Try that in the soft sciences.

It's still a frightening experience to read Alan Sokal's spoof "Transgressing the Boundaries: Towards a Transformative Hermeneutics of Quantum Gravity" from 1996 and follow the whole controversy it engendered.

Truth in Mathematics

Mathematics is about discovering truth in a certain abstract universe – that somehow relates to the real world, but it's difficult to say exactly what this relationship is. In the 19th century physics was a direct motivating factor for much the work in mathematics and the connection seemed fairly obvious, but in the age of string theory and multiple universes the physics link is now a bit tenuous. But maybe information theory is really the key to understanding reality.

At any rate, mathematical intuition guides us towards the truth and we have a sophisticated machinery available to make sure that we are indeed on track. Wrong claims inevitably seem to get crushed by this machinery (though it make take a few years for someone to discover the flaw in the argument).

Incidentally, the now standard cycle of definition, theorem and proof often leaves out and even obfuscates the discovery and experimental phase. Don't be fooled.

Weird Examples

Intuition versus Continuity

It became clear to mathematicians in the late 19th century that even well-honed intuition combined with established methods of proof (which were considered perfectly sufficient at the time) can sometimes lead one astray. The problem was that, as mathematics progressed to more and more abstract domains and increasing intellectual depth, intuition and common sense became less reliable.

Here is a classical example: the relationship between differentiable and continuous functions. It is clear the continuity is not enough for differentiability, a function can have "corners" such as the absolute value function at 0.

With a little bit of effort one can manufacture a continuous function that has countably many corners in the unit interval. E.g., one could have a kink at all positions $1/n$, n a positive integer.

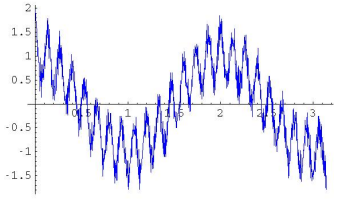
But that would seem to be about as bad as it can be.

Weierstrass's Monster

Alas, there are continuous functions that are nowhere differentiable. In a sense there are even more of them than of the other kind. Here is a notorious example:

$$f(x) = \sum_{n<\infty} b^n \cos(a^n \pi x)$$

For parameter values $0 < b < 1$ and $ab > 1 + 3/2\pi$ this function f is continuous but nowhere differentiable. One can almost see from a plot how the function works.

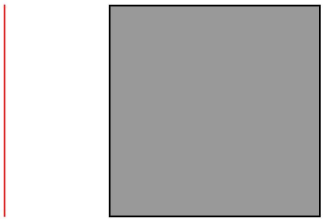


q

Comparing Sizes

Here is another example where common sense and intuition tend to lead one astray.

Which is larger, the red unit interval $[0, 1] \subseteq \mathbb{R}$ or the gray unit square $[0, 1]^2 \subseteq \mathbb{R}^2$?



Clearly the square is larger. In fact, we can place the the line segment inside the square in infinitely many non-overlapping ways. Hence, by ordinary common sense, the square must be much, much larger than the unit interval.

Right? Wrong!

Theorem. *Cantor (1845-1918)*

There is a bijection between the unit interval and the unit square.

And it does not end there. There is a bijection between the unit interval and the unit cube, the unit 4-dimensional hypercube, any n-dimensional hypercube whatsoever, and even all of n-dimensional Euclidean space. E.g., there is a bijection between $[0, 1]$ and $\mathbb{R}^{10^{10^{10}}}$.

In Cantor's own words:

> "Je le vois, mai je le ne croix pas."

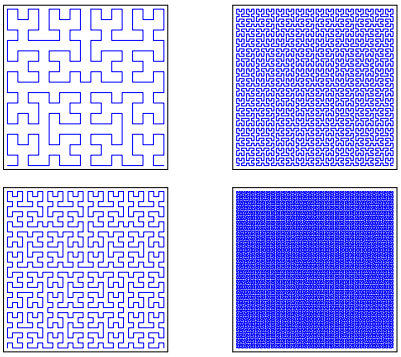
(I see it, but I can't believe it.) Thus, his intuition was not easily reconciled with the result of his argument.

Many of his colleagues rejected Cantor's ideas out of hand, but they are now the foundation of mathematics. His method of diagonalization is essential for computability and complexity theory.

Hilbert Curve

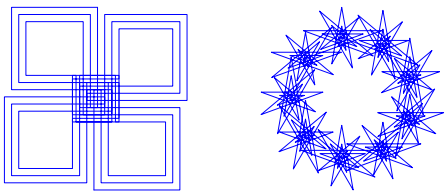
Cantor's theorem can be proven very abstractly using his machinery of cardinal numbers. There is also an argument based on doodling, due to Hilbert, the leading mathematician around 1900.

A Square-Filling Curve



Frivolous Digression

The pictures of the Hilbert curve were made using a stack turtle (a turtle on steroids).



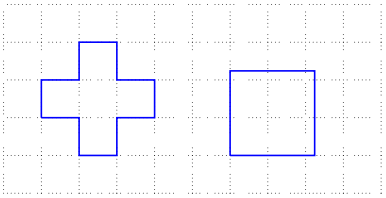
The code is on the course website.

Triangulating Polygons

One more example where intuition fails. First, a result in plane geometry that is perfectly common sense and not particularly surprising.

Theorem. *Bolyai-Gerwin Theorem*

Let P and Q be two polygons of equal area. Then P can be cut into finitely many triangles that can be reassembled to form Q.



The Enron Soccer Ball

Naturally one would assume that a similar claim holds in 3-dimensional space.

Theorem. *The Banach-Tarski Paradox*

One can cut up a soccer ball into finitely many pieces and reassemble them to get another ball the size of the sun.

Don't take "cut up" too literally here, we have to admit much more complicated decompositions than in the 2-dimensional case.

In fact, this theorem uses an assumption, the so-called **Axiom of Choice** that's a bit problematic – but nowadays mathematicians generally accept AC without much discussion. Several important results depend on AC, life without it is quite tedious.

No Way

There are many intuitive objections to the Banach-Tarski Paradox.

A very popular one is: if you decompose a sphere of volume, say, 1 into pieces and reassemble the pieces, they still produce a ball of volume 1, so we must get back the same old sphere.

As it turns out, the pieces used in the decomposition don't have a volume: They are so bizarre that one cannot measure their volume. Hence the invariance principle for volume is not violated.

See: Stan Wagon, "The Banach-Tarski Paradox", CUP 1985.

Insulting the Fathers

Of course, it takes a bit of effort to construct such pathological examples and to establish their properties. Moreover, they do not seem to play much of a rôle in the "real world". In fact, the relevance of pathological objects such as the Weierstrass function is not beyond dispute. Here is the response of one of the giants of the field.

> Formerly, when one invented a new function, it was to further some practical purpose; today one invents them in order to make incorrect the reasoning of our fathers, and nothing more will ever be accomplished by these inventions.
>
> H. Poincaré

Still, one cannot simply ignore such strange objects without risking a serious weakening of the foundations. And when we look to applications of mathematics to computer science rather than physics another layer of reliability is stripped away.

Intuition versus Proof

So: a careful formal presentation of an argument may help us to avoid mistakes where pure intuition might fail and results may be a bit counterintuitive.

Disclaimer:

This is not to say that intuition is not of central importance. In fact, without a strong intuition nothing moves, nothing whatsoever: finding the content of an interesting theorem, the assertion itself, does not appear to be a process that can be formalized.

All we claim is that in mathematics as well as computer science it is good to perform a soundness check, to make sure that in fact all the steps in the argument really hold water.

Incidentally, the tools we are going to develop can actually help sharpen one's intuition quite a bit.

Physics

It is noteworthy that the pitfalls of simple reliance on intuition and common sense are very well understood in physics, and have been for about century.

Suppose someone shoots an arrow at you, moving at, say, $100\,m/s$. If you were to run away from the arrow at $50\,m/s$ the impact velocity would be reduced to $50\,m/s$, if you could run at $99\,m/s$ the arrow would strike at a harmless $1\,m/s$. This is all entirely clear and standard Newtonian physics.

Now suppose you are being fired at with a laser. Instead of arrows you have to dodge photons moving at the speed of light c, approximately $299792458\,m/s$. But there is a peculiarity about the speed of light, it is invariant under movement of the frame of reference. So, even if you were to run from the laser beam at $9/10\,c$, it would still hit you at the ordinary speed of light.

This apparently paradoxical claim is well-established and in perfect keeping with experimental evidence. Similarly wild problems appear in quantum physics.

Programming versus Physics

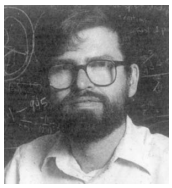
In physics the general theory of relativity and quantum theory provide a framework to deal with a host of bizarre and unexpected phenomena. Their methods are exceedingly successful and both theories have been confirmed in countless experiments and applications. Still, both theories are very counterintuitive and it takes some effort to learn how to use them.

So how about programming (never mind other activities in computer science)?

Relativity theory and quantum physics may be hard, but most people think that writing a piece of code is straightforward and requires nothing much beyond typing skills (OK, we exaggerate slightly for better effect).

For simple toy programs (insert an element at the end of a list, compute the sum of two fractions in reduced form) there may be some truth to this, but this is like saying that everyone can understand Hooke's law in physics, so physics must be easy.

Computer Science is Hard



> The standard of correctness and completeness necessary to get a computer program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.
>
> William P. Thurston

B. 1946, Fields Medal 1982

Formalizing Reasoning

Faulty Proofs

One consequence of the existence of pathological objects is that some "theorems" may not be quite true as stated (more hypotheses are needed, the claim needs to be weakened, the definitions need to be tightened . . .).

But what about the proof that accompanies the theorem?

Well, proofs can be very, very complicated. Moreover, even an apparently rigorous proof may have holes and errors that are quite difficult to spot.

In particular when the underlying concepts are not perfectly clearly defined and/or have counterintuitive properties the argument may seem perfectly reasonable when in fact it is subtly incorrect.

At issue here are not simple mistakes (such as forgetting a case or a computational error) but problems that hide deep inside the fabric of the argument.

Insuring Correctness

How can one guard against such problems?

- Use a special language to express the assertions.
- Use a deductive system to organize the logic of the argument.
- Make all underlying assumptions explicit in axioms.

Note that this program is assumes that the basic mathematical machinery is fully intact, one just needs to be a bit more careful in applying it.

The language typically used in this endeavor is predicate logic (though there are other, more complicated frameworks).

Together with a few well-chosen axioms that describe rock solid truths and some simple rules of logical inference one has a system that, though slightly cumbersome to use in the real world, is guaranteed to produce only correct results.

Truth has been translated into proof in a certain logical system.

Formalizing Reasoning

In order to be able to argue about assertions without steady appeals to intuition and indeed have the arguments be carried out by programs we need to

- formalize the statements (precise definitions, make tacit assumptions explicit), and
- formalize the logic, the process of reasoning (make every single proof step explicit, mechanical, checkable).

Upon closer inspection we actually need three things:

- a *foundational system* in which we can describe our domain of discourse,
- a *logic* that explains how to reason,
- a *formal language* in which to express all this stuff.

Brief History

Historically, logic as a purely mathematical discipline came first: an attempt to understand human reasoning, in particular as it relates to science. Foundational issues in mathematics first arose around late in the 19th century, and applications to computing in the last quarter to the 20th century.

- Aristotle (384-322): syllogisms
- G. Leibniz (1646-1716): *calculus ratiocinator*
- G. Boole (1815-1864): propositional calculus
- G. Frege (1848-1925): Begriffsschrift, set theory
- G. Peano (1858-1932): Peano Arithmetic
- B. Russell (1872-1970): Principia Mathematica
- E. Zermelo (1871-1953): set theory
- D. Hilbert (1862-1943): proof theory
- K. Gödel (1906-1978): incompleteness
- A. Tarki (1902-1983): truth, decidability
- D. Scott (1933-): semantics of programming languages

Implementation

Since we are interested in algorithms, we need to be a bit more careful than if we were simply explaining the basic ideas in logic abstractly, without a view towards implementation and computation. Getting one's intuition right is always the first step, but may not be enough.

As a result, our presentation may seem on occasion overly fastidious, rigorous and formal, but always bear in mind Thurston's comment.

Without a certain amount of intellectual pain, the program won't run in the end.

Or it may run only on occasion, produce wrong results, . . .

On the other hand, if things are correctly implemented we obtain very powerful tools that can help tackle problems that would otherwise be hopeless.

Logic and Computer Science

Logic is usually categorized into

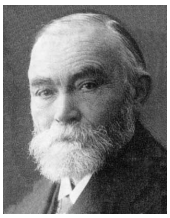
- Set theory
- Model theory
- Proof theory
- Recursion theory

It is clear that recursion theory, the theory of computability is relevant to CS, but it turns out that the other branches are all important, too. The importance lies less in the immediate results than the techniques used in these areas.

In fact, one can argue that logic relates to computer science in much the same way calculus relates to physics.

Frege's System

Frege's Work



Gottlob Frege (1848-1925) was the first to tackle the problem of constructing a rigorous deduction system that, he hoped, ultimately would be capable of expressing all of mathematics.

Frege understands very clearly that some assumptions will remain without proof but insists that these are spelled out clearly as axioms. He also is the first to explicitly define the logical rules used in his system.

Frege

He criticizes others (Dedekind, Peano) for the lack of precision in their work.

> It cannot be demanded that everything be proved, because that is impossible; but we can require that all propositions used without proof be expressly declared as such, so that we can see distinctly what the whole structure rests upon. After that we must try to diminish the number of primitive laws as far as possible, by proving everything that can be proved. Furthermore, I demand – and in this I go beyond Euclid – that all methods of inference employed be specified in advance.

This is from "Grundgesetze der Arithmetik I", published in 1893.

In 1879 Frege published "Begriffsschrift", which translates roughly as "concept script". Here he establishes a language and a (very powerful) logic, the groundwork for the Grundgesetze.

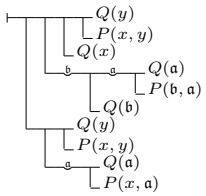
Begriffsschrift

Frege introduces a two-dimensional notation system – which, unfortunately, turns out to be very difficult to use.

The logical connectives implication and negation, as well as universal quantification are indicated by horizontal and vertical lines like so.

This may seem harmless, but if one constructs larger Formulae from these primitives things start to look quite ominous.

A Monster



Translated

In modern notation (a HUGE improvement):

$$\begin{aligned}[\forall a(P(x,a) \to Q(a)) \to (P(x,y) \to Q(y))] \to \\ [\forall b(Q(b) \to \forall a(P(b,a) \to Q(a)))] \to \\ Q(x) \to \\ P(x,y) \to Q(y)\end{aligned}$$

There are 4 premises, and altogether they imply $Q(y)$.

Frege's Logic

There is only one rule of inference, the *modus ponens*:

$$\frac{A, A \to B}{B}$$

The axioms for the propositional part, in modern notation, are

$$\begin{aligned} A &\to (B \to A) \\ (A \to (B \to C)) &\to ((A \to B) \to (A \to C)) \\ (A \to (B \to C)) &\to (B \to (A \to C)) \\ (B \to A) &\to (\neg A \to \neg B) \\ \neg\neg A &\to A \qquad A \to \neg\neg A \\ x = y &\to (P(x) \to P(y)) \qquad x = x \end{aligned}$$

Quantification

Quantification is handled by the (nowadays standard) axiom

$$\forall x A(x) \to A(t)$$

From a modern point of view this may seem all straightforward, but at the time (in 1879) this was completely new and unheard of.

A central goal of Frege was to use his machinery to derive numbers from within a purely logical framework.

Alas, Frege was ahead of his time; reviews of his work (by luminaries such as Cantor, Schröder, Peano) were usually unfavorable. In fact, Peano found it necessary to withdraw part of his review after receiving a detailed criticism thereof from Frege.

Objects and Concepts

Frege distinguishes between

- *Objects:* all the items in the domain of discourse.
- *Concepts:* properties, characteristics, attributes. An object may "fall under" a concept of fail to do so.
- *Extensions:* objects associated with concepts, very much like the "set of all objects with property such and such".

Frege's intent was to deal with all of the "real world" without any restrictions.

Quantification over concepts is allowed, so one can define equality:

$$a \approx b \iff \forall X\,(X(a) \to X(b)).$$

This is Leibniz's principium identitatis indiscernibilium.

Grundgesetze

In his Grundgesetze he added axioms to deal with sets (collections of objects). In essence he proposes an *Axiom of Extensionality* which, in modern notation, would look like so:

$$x = y \text{ iff } \forall z\,(z \in x \iff z \in y)$$

Two sets are the same iff they contain precisely the same elements.

A minor consequence of this axiom is that in sets order and multiplicity do not matter:

$$\{1, 2\} = \{2, 2, 1, 2, 2, 1, 1, 1\}$$

As a consequence, (finite) sets are actually harder to implement than nested lists.

More Extensionality

More importantly, the *description* of the set is irrelevant, only its extension matters.

$$A = \{1, 2\}$$
$$B = \{\, n \in \mathbb{N} \mid x^n + y^n = z^n \text{ has pos. integer solution} \,\}$$

Then $A = B$.

But this is Fermat's Last Theorem, the proof covers many hundreds of pages, took some 350 years to find, and uses very, very deep results.

So equality between sets may be very difficult to check even if one of them is entirely trivial and explicitly given.

Formation Axiom

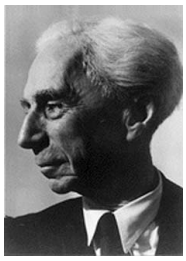
Another crucial axiom, going back to Cantor's ideas, regulates the formation of sets: for every predicate $P(z)$ there exists a corresponding set. In other words, any set you can ever think of really exists.

$$\exists\, x\, \forall\, z\, (z \in x \iff P(z))$$

The quantifiers all range over the collection of all sets.

Note that by Extensionality the set x in Formation is unique: any other x' would have the same extension as x and thus x and x' are identical.

Russell's Paradox



Unfortunately, Frege made a fundamental mistake: his axioms are self-contradictory.

This is rather surprising since Frege's system was an attempt not to create some bizarre theory (as seems popular now in physics) but to spell out the details of the reasoning process in mathematics.

B. Russell pointed the inconsistency out in a letter to Frege in 1902, just when the second volume of his Grundgesetze was about to go to print. Frege acknowledged the problem in an appendix, and offered a solution, but it turned out a few years later that the solution reduced the universe to size 1.

Russell's Set

The set discovered by Russell that shows that Frege's axioms don't quite make sense is annoyingly simple to state. Consider

$$S = \{\, z \mid z \notin z \,\}$$

S looks strange, but why not? It exists by Formation, and is unique by Extensionality.

But then both $S \in S$ and $S \notin S$ lead to a contradiction, and we have a disaster at our hands.

While it is intuitively clear that Russell's set is artificial and will not appear in, say, a discussion of Fourier transforms, it is rather difficult to fix this problem. All known solutions require some technical acrobatics: Zermelo-Fraenkel set theory, Russell type theory. Nowadays Zermelo-Fraenkel (with the Axiom of Choice) is generally accepted as the de facto foundation of Mathematics (in part due to Bourbaki, though one might argue that category theory will ultimately turn out to be a better approach).

Principia Mathematica

From 1910 till 1913 Russell together with his colleague A. Whitehead published 3 volumes of *Principia Mathematica*, a gargantuan effort to reconstruct a large piece of Mathematics in a logic framework that avoids problems like Russell's paradox (Russell's type theory) while at the same time adhering to the same level of precision as Frege.

The notation there is better than Frege's, but it's still hard to read.

Largely ignored by mathematicians, but type theory has become very important in logic and computer science.

Russell later complained that his intellect

> ". . . never quite recovered from the strain of writing it."

Zermelo-Fraenkel

As already pointed out, a version of set theory due to Zermelo and Fraenkel serves as the de facto foundation for all of mathematics at present.

One major departure from Frege's system is that set formation is much more restricted; we now have principles like the Aussonderungsaxiom (comprehension axiom):

$$\exists\, x\, \forall\, z\, (z \in x \iff z \in a \wedge P(z))$$

Thus, we can only construct a subset of an already given set a using some arbitrary predicate P. This, together with a list of other set-existence axioms seems to be perfectly enough to build all the sets needed in mathematical discourse.

And, at least on the face of it, these restrictions dismantle Russell's paradox. Alas, consistency proofs of set theory are based on acts of faith more than just unassailable reasoning.

Concepts versus Implementation

One can think of ZF set theory, usually augmented by the Axiom of Choice to ZFC, as a universal environment in which all (or at least almost all) concepts of mathematics and computer science can be implemented.

To make sure that some concept is indeed solid and precise give a definition in ZFC. For example, natural numbers, rationals, reals, limits, relations, functions, graphs, finite state machines, Turing machines, computable functions and so on can all explained in this way.

But note: a definition in ZFC is not necessarily the best way to understand the ideas behind it, the actual intent and meaning, much less the importance of the concept.

It is crucially important to have this more intuitive, informal understanding along with the precise definitions.

Deduction Systems

Modeling Proofs

Frege's and Russell's system are very ambitious in that they attempt to provide a foundation for all (most?) of mathematics. Clearly, one may get better results by focusing on more restricted domains of discourse instead.

There are several different kinds of formal systems that turn out to be useful in various contexts, to name but a few:

- Propositional Logic
- Equational Logic
- Predicate Logic
- Modal Logic
- Computation Tree Logic

All of these model a certain aspect of reasoning by giving exact rules of inference that can be used to derive theorems in the system.

Here is one example of such a collection of inference rules, for predicate logic.

Gödel Style Proofs

Informally, a *Gödel style proof* is a sequence of assertions, each of which is either justified by earlier ones, or simply taken for granted.

The assertions taken for granted are often called *axioms*. Axioms are arbitrary in a sense, though in real applications they are chosen very carefully and encapsulate the salient features of the domain of discourse.

It was one of Frege's accomplishment to realize that the underlying language and the rules of inference must be explicitly specified, otherwise it becomes difficult to determine whether proofs are indeed correct.

The axioms and rules should be sound (no contradictions can be derived) and complete (all "valid" assertions can be derived).

Soundness is essential, but completeness is often unattainable (and, in an algorithmic context, even undesirable).

Formal Proofs

Suppose we have fixed a formal system. Consider a set of formulae Γ (possibly empty).

Definition 1. *A proof or derivation of a formula ψ in this system from Γ is a sequence*

$$\varphi_1, \varphi_2, \varphi_2, \ldots, \varphi_{n-1}, \varphi_n = \psi$$

of formulae such that

- *φ_i is a logical axiom of the system, or*
- *φ_i is a formula in Γ, or*
- *φ_i follows from some of the φ_j, $j < i$, by some rule of inference (such as modus ponens).*

φ is said be provable from Γ (in the system), written $\Gamma \vdash \varphi$.

Automatic Theorem Provers

Exploiting Formal Systems

One important feature of formal systems is that their formulae are purely syntactic objects, easily represented by parse trees. As such they can be manipulated by programs.

Here is a wish-list of programs.

- Proof Checker: Given a list of formulae as input, determine whether the sequence is a valid proof in the system.
- Prover: Given a formula as input, finds a proof for the formula in the system (or determine that no proof exists).
- Decision Procedure: Given a formula as input, determine whether the is valid.

Decision Procedures are really independent of formal systems (except for the need of a formal language to express the assertions). The algorithms used to determine the truth in some particular domain of discourse are often very sophisticated.

Undecidability

The axioms of any reasonable formal system are always easy to deal with, they can be generated in a straightforward way and it is easy to check if a given formula is an axiom. Likewise, the rules of derivation are just rewrite rules that require some amount of pattern matching and some syntactical manipulation of the formulae in question.

As a consequence, proof checkers are perfectly feasible given a system that is sufficiently expressive to capture the proofs in question.

Alas, provers and decision procedures are very, very difficult to build even in a limited domain of discourse. Efficiency is an enormous problem.

Needless to say, in general one brushes up against undecidability: the collection of all theorems of a formal system is semi-decidable but not decidable, and validity of a formula is not even semi-decidable. For example, truth in elementary number theory is not even arithmetic.

The Hard Truth

While proof checkers are feasible as a matter of principle, the required formalization of proofs is really an unattainable ideal for most areas.

First off, mathematics is fundamentally undecidable, hence there must be very large proofs that are impossible to manipulate.

Second, even if the proofs are of feasible size, reconstructing all of mathematics in a formal setting is a daunting task at best.

There is also a clear and present danger that such an effort would lead to unintended and most undesirable side-effects: see the work of Bourbaki.

See

`http://mizar.org`

for an attempt to capture (a lot of) mathematics in a verifiable formal system.

Humans versus Machines

There is a fundamental dilemma when it comes to the presentation of proofs: we can either

- insist on human-readability so some theorems will never be proven, or
- accept machine-readable proofs and unravel the social fabric of math.

It is unclear whether many of the theorems one would miss out on by insisting on "humane" proofs are of any real interest, but it is hard to see that all of them could be irrelevant.

Classical proofs are supposed to generate insight into the nature of the problem beyond just establishing the truth of an assertion; it would seem that this additional feature would have to be abandoned if we allow machine-only-readable proofs.

Famous Examples from Mathematics

1966 Lander, Parkin: counterexample to Euler's conjecture

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

1976 Appel, Haken: Four Color Theorem

1997 McClune, Wos: Robbin's Conjecture

1998 Hales: Kepler's Conjecture

2006 Gonthier: Verification of Four Color Theorem in Coq

Again, some of these results were/are quite controversial in the mathematics community.

Kepler's Conjecture

There is an intuitively obvious way to stack balls of equal size so as to minimize the amount of space required.

Kepler stated in 1611:

The packing will be the tightest possible, so that in no other arrangement could more pellets be stuffed into the same container.

The proof by Tom Hales in 1998 was a fundamental result, yet the Annals of Mathematics refused to publish it without a disclaimer. After 4 years of review, the referees were only "99% certain" that the result is correct.

Read `http://www.math.pitt.edu/~thales/flyspeck/index.html` for more information.

Flyspeck Project

Responding to the unkind reception in the mathematics community of his result, Hales decided to grab the bull by its horns.

The purpose of the flyspeck project is to produce a formal proof of the Kepler Conjecture, using the full arsenal of automated theorem proving.

All computer proofs will have correctness guarantees.

The project has created widespread engagement in the ATP community.

Flyspeck has already produced a proof of the Jordan curve theorem in HOL-light. Alas, the proof is hard to read.

Incidentally, Hales states that "Mathematica has been used extensively throughout this project for experimentation and exploration."