ADVENTURES IN FOUNDATIONS OF MATHEMATICS

Ross Program 2022
MWF during June 27 - July 8, 2022
1. Logical reasoning
2. Finite set theory
3. Infinite sets and ZFC
4. Order equivalence, emulators, and stability
5. Stability of emulators of two pairs
6. Stability of emulators of three pairs

Harvey M. Friedman

1. LOGICAL REASONING

revised June 28, 2022

We can think of mathematics as a skyscraper. In the Ross program you work in the middle of the skyscraper. This is because understanding the upper levels usually require some years of mastery at middle levels, and so Ross can't do much with the upper levels. On the other hand, the lower levels are actually rather subtle and involved and you could get trapped there and not move up even to the middle levels that Ross thrives in.

The first week of this 6 lecture series presents the basement of mathematics. The material that I will cover in the first week (first three lectures) needs a full semester undergraduate math course, and usually revisited in greater depth in a full semester graduate math course.

There is little experience at Ross with working with students in the basement. So in a way this is a Ross experiment.

Logical reasoning is a rather rich and subtle matter, and it really should be looked at as something that is even lower than the basement of mathematics! It is really pre-mathematics. Many of you will find this first lecture as the furthest away from your mathematical experiences at Ross.

I really have only one hour to present this, which I wouldn't even consider attempting except for the exalted reputation of Ross students! But it is essential for an understanding of the basement of mathematics.

Things will let up and become more comfortable and familiar in the second lecture on Finite Set Theory. Here we teat finite set theory as a traditional mathematical subject but with no algebraic or geometric structure. This should be nice and friendly.

The third lecture extends Finite Set Theory to Infinite Set Theory, and then combines this with the Logical Reasoning in this lecture to form the standard foundations of mathematics, ZFC.

Armed with this in depth look at the basement of mathematics in the first week, you will have enough background to place the second week in context. The material in the second week is of a strictly combinatorial nature, with no algebraic or geometric structure. It lives in dimension 2, where it is already rather rich and subtle. In higher dimensions we know that it actually "breaks" the longstanding ZFC foundations, and this work is being finished up now.

Now let's look at the structure of the basement.

1. prop logic lifts to pred logic
2. finite set theory lifts to infinite set theory
3. pred logic and infinite set theory generates ZFC

ZFC = Zermelo Frankel set theory with the Axiom of Choice.

Propositional and predicate logic are used all the time in mathematics, especially when writing proofs. However they are also used in careful thought having nothing particularly to do with mathematics.

It has proved to be very clarifying and useful to separate this logic from mathematics and study it and develop it in its own right. This turned out to be a very good idea because much later it came to be very useful and clarifying in theoretical and applied computer science.

So let's start with propositional logic. There is a very basic way of combining two statements into a third so the meaning of the third is readily understood just from the meaning of the two. For example,

every 2022 Ross student is a nice person
AND
some 2022 Ross student is over 6'6" tall

Propositional logic doesn't care about anything here except the word "AND". It doesn't care that "nice person" has never been reasonably defined and that 6'6" has been, and it doesn't care

about those very important precise words "every" and "some". It only cares about AND.

NOTE: Propositional logic doesn't care about "every" and "some". But the deeper predicate logic does!

Let's talk about this situation like a mathematician. What we are looking at can be written as

p AND q

where we use p,q to stand for unknown statements, just like we use letters for unknowns in algebra.

Propositional logic only cares about whether

p is true or false
q is true or false
p AND q is true or false

Actually propositional logic has no idea about which alternative, true or false, in the first two. Just like algebra doesn't know what x,y are when we just write x+y.

But propositional logic knows and cares about how the third is affected by the first two. This dependence is what propositional logic is all about. We can write this dependence by a table.

p	q	p AND q
T	T	T
T	F	F
F	T	F
F	F	F

Notice that there are four rows and if we look at what is under columns p,q, we see what are called the truth assignments for p,q. Also the truth assignments for p AND q.

So each row gives us the truth assignment and the truth value.

AND is called a Propositional Connective. There are five connectives used over and over again in mathematics. AND, OR, IMPLIES, IFF, and also NOT. NOT is simpler in that it only operates on a single statement. For instance, if we have any A we can form NOT A. The truth table for NOT is

p	NOT p
T	F
F	T

OR is also very intuitive, but of course quite different.

p	q	p OR q
T	T	T
T	F	T
F	T	T
F	F	F

IFF = if and only if is maybe not as obvious but still pretty intuitive.

p	q	p IFF q
T	T	T
T	F	F
F	T	F
F	F	T

The idea behind IFF is "being equivalent".

We now come to the one causing the most confusion.

p	q	p IMPLIES q
T	T	T
T	F	F
F	T	T
F	F	T

It is the last line with the truth assignment p = F, q = F, that causes the most confusion. But experience shows that propositional logic with these truth tables are essential in foundations of mathematics, and therefore we have to use T or F at the right corner. The idea is that once we know p is false we shouldn't care about q.

Another way of looking at this truth table for IMPLIES is that p IMPLIES q is wrong if and only if p is true and q is false. That corresponds to the only row ending with F is the second row.

Just as in algebra, we can make compound expressions, and these will also have their own truth table. I'll do two illustrative examples.

p	q	r	q ∧ r	p ∨ (q ∧ r)
T	T	T	T	T
T	T	F	F	T
T	F	T	F	T
T	F	F	F	T
F	T	T	T	T
F	T	F	F	F
F	F	T	F	F
F	F	F	F	F

p	q	r	p ∨ q	p ∨ r	((p ∨ q) ∧ (p ∨ r))
T	T	T	T	T	T
T	T	F	T	T	T
T	F	T	T	T	T
T	F	F	T	T	T
F	T	T	T	T	T
F	T	F	T	F	F
F	F	T	F	T	F
F	F	F	F	F	F

Note that this time we used three letters so there are eight rows for the eight truth assignments. But look at this! The last columns are the same!! This means that those two compound formulas have the same truth values under the same truth assignments! This fact is written

$$p \vee (q \wedge r) \equiv ((p \vee q) \wedge (p \vee r))$$

with ≡ read "logically equivalent".

We now get more mathematical about what we have been doing. Propositional logic uses infinitely many letters (variables). Instead of the capital letters we have been using, we use $p_1, p_2, \ldots$, indexed by positive integers.

Propositional formulas are defined inductively as follows.

i. Every p_i is a prop formula.
ii. If A,B are prop formulas then so are $\neg A, A \wedge B, A \vee B, A \rightarrow B, A \leftrightarrow B$.
iii. Prop formulas only arise from i,ii.

NOTE: There is a problem with ambiguity. E.g., consider $\varphi \vee \psi \wedge \rho$. Is it $(\varphi \vee \psi) \wedge \rho$ or is it $\varphi \vee (\psi \wedge \rho)$? Thus we really need to use parentheses in ii above. And there are conventions that reduce the number of parentheses required. All of this is supported by what are called "unique parsing theorems". This is an important

topic in computer science regarding formal languages, with associated clever efficient algorithms. We won't go into this here.

NOTE: What are inductive definitions and how are they justified? This is by no means trivial but is well understood and part of the official basement. But we don't have time to go into all of this here.

A truth assignment is a function that assigns a truth value (T or F) to every letter $p_1,p_2,\ldots$. I.e., an $f:\{p_1,p_2,\ldots\} \to \{T,F\}$.

(In the lecture I also considered partial truth assignments. I decided that this turns out to be more trouble than it is worth).

So we define SAT(A,f) inductively as follows. This is read "A is satisfied by f" or "A comes out T under f".

i. SAT(p_i,f) if and only if $f(p_i) = T$.
ii. SAT($\neg$A,f) if and only if not SAT(A,f).
iii. SAT(A$\wedge$B,f) if and only if SAT(A,f) and SAT(B,f).
iv. SAT(A$\vee$B,f) if and only if SAT(A,f) or SAT(B,f).
v. SAT(A$\to$B,f) if and only if not(SAT(A,f) and not SAT(B,f)).
vi. SAT(A$\leftrightarrow$B,f) if and only if (SAT(A,f) and SAT(B,f)) or (not SAT(A,f) and not SAT(B,f)).

You should see (at least feel) that this inductive definition corresponds exactly to what we were doing with truth tables, but where we only use the values of the truth assignment at letters that appear in the formula. The rows correspond to the truth assignments f at letters appearing in the formula. When column A meets row f we see T if SAT(A,f), and F if not SAT(A,f).

More generally, let K be a set of prop formulas and f be a partial truth assignment. SAT(K,f) means that for all A $\in$ K, SAT(A,f).

THEOREM 1. Let f,g be truth assignments that agree at all letters that appear in some element of K. Then SAT(K,f) if and only if SAT(K,g).

A prop formula A is said to be valid if and only if for all truth assignments f, SAT(A,f). This is the same as saying that in the truth table for A, the column under A is all T's.

A set K of prop formulas is said to be valid if and only if for all truth assignments f, SAT(K,f).

A prop formula A is said to be satisfiable if and only if for some truth assignment f, SAT(A,f). This is the same as saying that in the truth table for A, the column under A has at least one T.

A set K of prop formulas is said to be satisfiable if and only if for some truth assignment f, SAT(K,f). For K finite, this is not really new as we shall see in Theorem 2. For K infinite, it is something new.

THEOREM 2. Let $A_1,\ldots,A_n$ be prop formulas, $n \geq 1$. Then $\{A_1,\ldots,A_n\}$ is satisfiable (valid) if and only if $(\ldots(A_1 \wedge A_2) \wedge \ldots \wedge A_n)$ is satisfiable (valid).

THEOREM 3. A is valid if and only if $\neg A$ is not satisfiable. A is satisfiable if and only if $\neg A$ is not valid. K is valid if and only if no $\neg A$, $A \in K$, is satisfiable.

K is satisfiable if and only if what??

THEOREM 4. K is satisfiable if and only if every finite subset of K is satisfiable.

We have only letters $p_1,p_2,\ldots$ in prop logic. Prop logic has also been studied with any set of letters whatsoever. Like p_x, $x \in \Re$. Turns out that Theorem 4 still holds. However it is very different than the case $p_1,p_2,\ldots$ in that Theorem 4 cannot be proved with the p_x, $x \in \Re$ without using some form of the Axiom of Choice; i.e., it cannot be proved in ZF. We know that this is the case, because using tools from mathematical logic, we can actually prove this claim.

How can we tell if a prop formula is satisfiable? Make the truth table. Note that this has 2^n rows where n is the number of letters. However, this is computer intensive for, say, n = 30, and computer impossible for, say, n = 50. Is there a more efficient way to tell?

Let's actually be more exact about this problem. Is there a computer algorithm which does the following. The inputs are the propositional formulas A where the variables p_i are written with the subscript i in base 2. So A is a finite string in the ten letter alphabet $0,1,p,\neg,\wedge,\vee,\rightarrow,\leftrightarrow,(,)$. We want the algorithm to

determine whether A is satisfiable, where the running time is always at most a polynomial (fixed in advance) in the number of symbols in A.

If you can answer this question yes or no, submit your proof to win $1,000,000 and also instantly become the most famous living mathematician/computer scientist. This is really jumping to the top!

And your parents will definitely think that the Ross program was well worth the tuition!!

WARNING: Some very well known mathematicians think that this problem, which is well known to be equivalent to the so called P = NP problem in theoretical computer science (Stephen Cook), will be the last Millennium Prize Problem to be solved.

DEFINITION. A set K of prop formulas logically implies a prop formula A if and only if for all truth assignments f, if SAT(K,f) then SAT(A).

THEOREM 5. K logically implies A if and only if some finite subset of K logically implies A.

Here think of K as a set of premises, and A as the conclusion.

We would like to have a nice axiomatic system T for propositional logic with these properties.

1. If A can be proved in T then A is valid. This is called soundness of the system T.
2. If A is valid then A can be proved in T. This is called completeness of the system T.
3. If K can be proved in T then K is valid. This follows immediately from soundness.
4. If K logically implies A then K proves A in T. This is called relative completeness.

THEOREM 6. If 4 holds for finite K then 4 holds for all K. I.e., if 4 holds for finite K then T is relative complete.

Of the sound and relatively complete systems for prop logic, the ones with the simplest form are the so called Hilbert systems. These consist of a finite number of prop formulas called axioms. A proof in the system consists of a finite list of prop formulas, where each entry

i. is a substitution instance of one of the axioms; or
ii. follows from some two previous entries by Modus Ponens.

For substitution instances, we replace letters by prop formulas, the same letter always being replaced by the same formula. Modus ponens is the rule: from A and A → B, derive B.

A proof of A is of a proof whose last entry is A. A proof of A from K consists of a finite list of prop formulas, where each entry

i. is a substitution instance of one of the axioms; or
ii. follows from some two previous entries by Modus Ponens; or
iii. is an element of K.

Most authors simplify matters and allow only the connectives ¬,→ in propositional logic. This is justified because we can treat each of the three remaining connectives, ∧,∨,↔, as abbreviations. I.e., A ∧ B for ¬(A → ¬B), A ∨ B for ¬A → B, A ↔ B for ¬(A → ¬B) ∧ ¬(B → ¬A).

NOTE: There are also some good reasons NOT to allow these abbreviations and treat the three remaining connectives directly.

This famous choice of three formulas yields a relatively complete system:

A → (B → A)
(A → (B → C)) → ((A → B) → (A → C))
((¬A) → (¬B)) → (B → A)

This is the "third Łukasiewicz axiom system" and appears along with many others on the Wikipedia page: List of Hilbert's Systems. See the Supplemental Material for handling all of the connectives.

There has been considerable interest in different kinds of relatively complete axiom systems for propositional logic. Most recently are the systems residing inside what are called Proof Assistants. There the systems need to conveniently accommodate how mathematicians really think when they write proofs. The idea is to allow the mathematician to make any reasonable moves that are seen in actual proofs. This has been quite successful, although it does move in the direction of engineering and not

theory. I have my own thoughts on this, but we cannot discuss it here.

Propositional logic is just not powerful enough to really be adequate for the logic of mathematics. It needs to be strengthened with additional essential logical features of mathematics.

Even ordinary commonsense reasoning lies outside the scope of propositional logic. Consider this famous example from Aristotle:

All men are mortal.
Socrates is a man.
Therefore Socrates is mortal.

The first two lines are the premises, and the third line is the conclusion.

The use of "all" goes beyond propositional logic in an essential way. And in order to support "all" we also need what are called predicates. Hence the name predicate logic.

We can break this valid inference down as follows. The predicates are "being a man", "being mortal", and we also have a naming feature, namely naming a particular object Socrates.

In predicate logic this is written

$(\forall x)(man(x) \rightarrow mortal(x))$.
$man(Socrates)$.
Therefore $mortal(Socrates)$.

These two predicates are one place or unary predicates. However, we all know the great importance in mathematics of two place predicates. Like $x < y$ and like $x \in A$.

It took nearly 2,300 years for us to fully realize that mathematics needs two place (and more) predicates and to incorporate this into logic. From Aristotle around 400BC to Frege around 1900.

So here is the full list of ingredients for predicate logic.

1. Variables. $v_1, v_2, \ldots$.
2. Constants. $c_1, c_2, \ldots$.
3. Relations. R^n_m, $n,m \geq 1$. R^n_m is n-ary.

4. Functions. F^n_m, $n,m \geq 1$. F^n_m is n-ary.
5. Quantifiers. $\forall$, $\exists$.
6. Equality. =.
7. Connectives. Usual. $\neg,\wedge,\vee,\rightarrow,\leftrightarrow$.

Need to define the formulas like prop formulas, and something corresponding to truth assignments.

For the formulas, the plan is

a. Define the terms. These are the expressions that refer to objects.
b. Define the atomic formulas.
c. Define the formulas.

TERMS

i. Variables and constants are terms.
ii. If $t_1,\ldots,t_n$ are terms then $F^n_m(t_1,\ldots,t_n)$ is a term.

ATOMIC FORMULAS

i. If s,t are terms then s = t is an atomic formula.
ii. If $t_1,\ldots,t_n$ are terms then $R^n_m(t_1,\ldots,t_n)$ is an atomic formula.

FORMULAS

i. All atomic formulas are formulas.
ii. If φ is a formula and v is a variable, then $(\forall v)(\varphi)$ and $(\exists v)(\varphi)$ are formulas.
iii. If φ,ψ are formulas then $\neg\varphi,\varphi\wedge\psi,\varphi\vee\psi,\varphi\rightarrow\psi,\varphi\leftrightarrow\psi$ are formulas.

On Friday, when we present the conventional foundation for mathematics, ZFC, we will be using this predicate logic, but with only one binary relation symbol ($\in$) and equality (=).

Remember we saw what it takes to determine whether a propositional formula is true or false. We used truth assignments to do that.

What does it take to determine whether a predicate calculus formula is true or false?

a. A nonempty set D of objects. This tells us what $\forall x$ and $\exists x$ means. For all $x \in D$, there exists $x \in D$.
b. For each constant symbol we assign an element of D.
c. For each variable we assign an element of D.
d. For each n-ary relation symbol we assign an n-ary relation on D; i.e., a subset of D^n.
e. For each n-ary function symbol we assign an n-ary function from D into D; i.e., a function from D^n into D.

We don't need to say anything really about = and about the five connectives.

NOTE: What does = really mean? What does "not" really mean? What does "and" really mean? These questions don't have really interesting answers today. These questions lie BELOW THE BASEMENT, in the ground, and look to the Philosophy departments for struggling with such questions.

From a,b,d,e we form what we call a structure, written M = (D,c*s,R*s,F*s), where c*s are the assignments in b, R*s are the assignments in d, F*s are the assignments in e.

NOTE: Constant symbols, but not variables, get interpreted as part of the structure M. Relations and functions also get interpreted as part of the structure M.

The structure itself, written M = (D,c*s,R*s,F*s), is not enough, generally, to determine whether a formula is true. For instance, consider $R^1_1(v_1)$. Does this hold in M? From M we know what 1-ary relation R on D is, but we have no indication as to what element of D that v_1 refers to.

The missing information that we need is what elements of D the variables v_i refer to.

So let M = (D,c*s,R*s,F*s). An M-assignment is a function $f:\{v_1,v_2,...\} \rightarrow D$. Then we can determine whether any formula φ of predicate logic holds in M at the assignment f. I.e., SAT(M,φ,f).

Just as we did for propositional logic, we make an inductive definition of SAT(M,φ,f).

Actually, we must first handle terms. I.e., VAL(M,t,f), the value of term t in structure M at assignment f.

VAL(M,c_i,f) = c_i*.
VAL(M,v_i,f) = f(v_i).
VAL(M,F^n_m(t_1,...,t_n),f) = F^n_m*(VAL(M,t_1,f),...,VAL(M,t_n,f)).

Now we are ready to handle atomic formulas.

SAT(M,s = t,f) if and only if VAL(M,s,f) = VAL(M,t,f).
SAT(M,R^n_m(t_1,...,t_n),f) if and only if
R^n_m*(VAL(M,t_1,f),...,VAL(M,t_n,f)).

We now come to predicate logic formulas. The crucial clauses of course regard the quantifiers ∀,∃.

SAT(M,¬φ,f) if and only if not SAT(M,φ,f).
SAT(M,(φ∧ψ),f) if and only if SAT(M,φ,f) and SAT(M,ψ,f).
SAT(M,(φ∨ψ,f) if and only if SAT(M,φ,f) or SAT(M,ψ,f).
SAT(M,(φ→ψ,f) if and only if not(SAT(M,φ,f) and not SAT(M,ψ,f)).
SAT(M,(φ↔ψ,f) if and only if (SAT(M,φ,f) and SAT(M,ψ,f)) or (not SAT(M,φ,f) and not SAT(M,ψ,f)).
SAT(M,(∀v)(φ),f) if and only if for all x ∈ D, SAT(M,φ,f(v|x)).
SAT(M,(∃v)(φ),f) if and only if there exists x ∈ D, SAT(M,φ,f(v|x)).

Here f(v|x) is the same as f except that at v the value is x.

We say that φ is valid if and only if for all structures M and M-assignments f, SAT(M,φ,f). We say that φ is satisfiable if and only if there is a structure M and an M-assignment f such that SAT(M,φ,f).

These two definitions are repeated for sets K of formulas. K is valid if and only if for all M,f, SAT(M,φ,f). K is satisfiable if and only if there exists M,f such that SAT(M,φ,f).

THEOREM 7. Let A_1,...,A_n be formulas, n ≥ 1. Then {A_1,...,A_n} is satisfiable if and only if (...(A_1 ∧ A_2) ∧ ... ∧ A_n) is satisfiable.

THEOREM 8. A is valid if and only if ¬A is not satisfiable. A is satisfiable if and only if ¬A is not valid. K is valid if and only if no ¬A, A ∈ K, is satisfiable.

THEOREM 9. K is satisfiable if and only if every finite subset of K is satisfiable.

What about a complete system like we had for propositional logic? Recall this for prop logic:

$A \to (B \to A)$
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
$((\neg A) \to (\neg B)) \to (B \to A)$
From $A, A \to B$ derive B

Here this is for only the connectives $\neg, \to$. We build on this for predicate logic. Also using only the connectives $\neg, \to$.

1. $A \to (B \to A)$
2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
3. $((\neg A) \to (\neg B)) \to (B \to A)$
4. $v = v$
5. $v = w \to (X \to X')$
6. $(\forall v)(A) \to A[v/t]$
7. $A[v/t] \to (\exists v)(A)$
8. From $A, A \to B$ derive B
9. From $A \to B$ derive $A \to (\forall u)(B)$
10. From $A \to B$ derive $(\exists u)(A) \to B$

with a number of explanatory notes here.

A proof in this system consists of a nonempty finite sequence of formulas where every entry is an instance of 1-7 (with technical restrictions on 6,7), or follows from previous entries by 8,9,10 (with technical restrictions on 9,10).

Let K be a set of formulas. A proof in this system from K consists of a nonempty finite sequence of formulas where every entry is an instance of 1-7 (with technical restrictions on 6,7), or follows from previous entries by 8,9,10 (with technical restrictions on 9,10), or is an element of K.

Now some explanatory notes.

First of all, A[v/t] results from A by replacing all occurrences of the variable v in A by the term t.

Here A,B,C are arbitrary formulas and v,w are arbitrary variables. However, there is a technical restriction on variable u in 9,10, and a technical restriction on term t in 6,7.

In 5, formulas X,X′ are required to be atomic formulas, and X′ results from X by replacing zero or more occurrences of v by w.

Here are typical of the subtleties in 6,7. Look at

a. $(\forall v)(\exists w)(R(v,w)) \rightarrow (\exists w)(R(w,w))$.
b. $R(v,w) \rightarrow (\exists w)(R(w,w))$.

Both of these fall under 6,7 without restrictions, with t = w. But they are a disaster, clearly not valid.

Here is typical of the subtleties in 9,10. Look at

c. From $R(u) \rightarrow R(u)$ derive $R(u) \rightarrow (\forall u)(R(u))$.
d. From $R(u) \rightarrow R(u)$ derive $(\exists u)(R(u)) \rightarrow R(u)$.

These are a disaster as $R(u) \rightarrow R(u)$ is derivable even from just 1,2,3, and $R(u) \rightarrow (\forall u)(R(u))$, $(\exists u)(R(u)) \rightarrow R(u)$ are not valid.

The overall subtleties here are referred to as "clashes of variables". The requirement placed on 6,7 is that "t is free for v in A", and the requirement placed on 9 is "u is not free in A" and the requirement placed on 10 is "u is not free in B".

SUPPLEMENTARY MATERIAL

Prove Theorems 1-9.

Prove that a,b are both not valid.

Regarding c,d, show that $R(u) \rightarrow (\forall u)(R(u))$ and $(\exists u)(R(u)) \rightarrow R(u)$ are not valid.

If we use all of the connectives except $\leftrightarrow$, then this system will suffice (due to Kleene).

$A \rightarrow (B \rightarrow A)$
$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
$(A \wedge B) \rightarrow A$
$(A \wedge B) \rightarrow B$
$A \rightarrow (B \rightarrow (A \wedge B))$
$A \rightarrow (A \vee B)$
$B \rightarrow (A \vee B)$
$(A \rightarrow B) \rightarrow ((A \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))$
$(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$

¬¬A → A

To incorporate all of the connectives, add

(A ↔ B) → (A → B)
(A ↔ B) → (B → A)
(A → B) → ((B → A) → (A ↔ B))

The overall structure of these Hilbert systems are very simple, but as you can see, choosing adequate sets of axioms is far from trivial.

The most important and in many ways the most elegant of the relatively complete systems are the so called Gentzen systems. There are also Beth tableaux systems. These are both systems of propositional logic and predicate logic.

The approach taken with the so called Proof Assistants is to conveniently allow just about any logical move that mathematicians find convenient and intuitive to make. However, this everything and the kitchen sink approach has more the flavor of practical engineering than principled theory. I think there are some missed opportunities here for research.