Mathematical Logic and Computers

Some interesting examples

Michael Beeson

San José State University

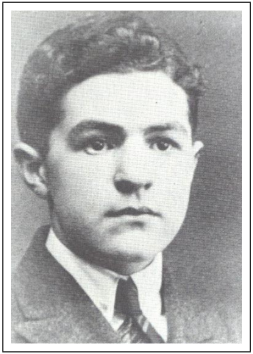
August 28, 2008

Early History

In 1954, within a few years after computers were first up and running, Martin Davis programmed the Presburger decision procedure for a vacuum-tube computer at the Institute for Advanced Studies in Princeton. That procedure, as is now known, has worse than exponential runtime, and according to Davis's recollections, "its great triumph was to prove the sum of two even numbers is even."

Mojzesz Presburger, 1904-1943

The young Presburger. He was a student of Alfred Tarksi, but died in the prime of life in a Nazi death camp. Not, however, before inventing Presburger arithmetic and giving a decision procedure for it.



Bertrand Russell was born too soon

- The same impulse that led Russell to write *Principia Mathematica* has led to the creation of general-purpose systems such as Automath, Coq, Isabelle, Mizar, and HOL-Light.
- Also Johan Belinfante has been using Otter to formalize set theory with a similar purpose.
- These applications of computers to the foundations of mathematics will not be discussed here.

What this talk is not about

- Use of logic as a language for computers to check mathematics, e.g. proofs of the Prime Number Theorem, Kepler's conjecture, or even Gödel's incompleteness theorem.
- Use of logic in computer design
- Use of computers to find (counter) models
- Model-checking to prove the correctness of hardware or software
- Theorem-proving to prove the correctness of hardware, software, or security protocols

Computerizing the metatheory

The applications of computers to mathematical logic that I will discuss fall under this framework:

- a formal system of logic, represented by formulas

Computerizing the metatheory

The applications of computers to mathematical logic that I will discuss fall under this framework:

- a formal system of logic, represented by formulas
- a proof predicate or a provability predicate, and/or other metamathematical predicates

Computerizing the metatheory

The applications of computers to mathematical logic that I will discuss fall under this framework:

- a formal system of logic, represented by formulas
- a proof predicate or a provability predicate, and/or other metamathematical predicates
- a metatheory for reasoning about these things

Computerizing the metatheory

The applications of computers to mathematical logic that I will discuss fall under this framework:

- a formal system of logic, represented by formulas
- a proof predicate or a provability predicate, and/or other metamathematical predicates
- a metatheory for reasoning about these things
- a theorem prover implementing the metatheory

Generality of this work

- Does not depend on a particular prover.
- The work reported was mostly done with Otter.
- It also works with Prover9.
- It will work with any prover that offers sufficient control over the basic search algorithms.
- It will still work fifty years from now with whatever provers are then in fashion.
- Maybe by then, it will be trivial for those provers.

Notation for implication

"A implies B" can be written as

- $A \rightarrow B$

Notation for implication

"A implies B" can be written as

- $A \rightarrow B$
- $A \supset B$

Notation for implication

"A implies B" can be written as

- $A \to B$
- $A \supset B$
- **C**AB (Polish notation, no parentheses)

Notation for implication

"A implies B" can be written as

- $A \to B$
- $A \supset B$
- **C**AB (Polish notation, no parentheses)
- $i(A, B)$ (suitable for theorem-proving)

Notation for implication

"A implies B" can be written as

- $A \rightarrow B$
- $A \supset B$
- **C**AB (Polish notation, no parentheses)
- $i(A, B)$ (suitable for theorem-proving)
- Example: $\mathbf{C}A\mathbf{C}BA$ is $i(A, i(B, A))$ or $A \supset (B \supset A)$.

Notation for other connectives

Negation is written $\sim$ or $\neg$, usually in prefix notation.
Łukasiewiczis credited with inventing "Polish notation" (named after his nationality).
He used **C** for implication, **A** for disjunction, **K** for conjunction, and **N** for negation.

Łukasiewicz's L1-L3

Łukasiewicz's system L1-L3

$$
\begin{array}{ll}
i(i(x,y),i(i(y,z),i(x,z))) & L1 \\
i(i(n(x),x),x) & L2 \\
i(x,(i(n(x),y))) & L3
\end{array}
$$

Frege's system

$$
\begin{array}{ll}
i(x,i(y,x)) & F1 \\
i(i(x,i(y,z)),i(i(x,y),i(x,z))) & F2 \\
i(i(x,i(y,z)),i(y,i(x,z))) & F3 \\
i(i(x,y),i(n(y),n(x))) & F4 \\
i(n(n(x)),x) & F5 \\
i(x,n(n(x))) & F6
\end{array}
$$

Church's system

$$
\begin{array}{ll}
i(x, i(y, x)) & C1 \\
i(i(x, i(y, z)), i(i(x, y), i(x, z))) & C2 \\
i(i(n(x), n(y)), i(y, x)) & C3
\end{array}
$$

Hilbert's system

$i(x, i(y, x))$	$H1$
$i(i(x, i(y, z)), i(y, i(x, z)))$	$H2$
$i(i(y, z), i(i(x, y), i(x, z)))$	$H3$
$i(x, i(n(x), y))$	$H4$
$i(i(x, y), i(i(n(x), y), y))$	$H5$
$i(i(x, (i, x, y)), i(x, y))$	$H6$

Detachment and substitution

- *modus ponens*
  From A and $i(A, B)$, infer B.

Detachment and substitution

- *modus ponens*
  From A and $i(A,B)$, infer B.
- *substitution*
  From B infer $B\sigma$, where σ is a substitution, that is, a function from variables to terms.

Detachment and substitution

- *modus ponens*
  From A and $i(A, B)$, infer B.
- *substitution*
  From B infer $B\sigma$, where σ is a substitution, that is, a function from variables to terms.
- *detachment*
  From A and $i(B, C)$, where substitution σ makes $A\sigma = B\sigma$, infer $C\sigma$.

Example of Detachment

$i(i(n(x), x), x))$ the major premise $i(B, C)$
$i(n(n(x), n(x))$ the minor premise A
Rename variables in minor premise $i(n(n(y), n(y))$
Unify B, which is $i(n(x), x)$, with A. The result is the substitution $x : n(y)$. Conclusion $n(y)$. We can rename the variable to x again if we like.

Relations between MP, Substitution, and detachment

- Modus ponens is a special case of detachment (with the identity substitution)
- Once $i(x, x)$ has been deduced, substitution is also a special case of detachment.

Łukasiewicz used the rules of detachment and substitution. Meredith later introduced the rule of *condensed detachment*, which is detachment with the further requirement that σ be the most general unifier of A and B, rather than just *some* unifier.

Scott's Challenge of 1990

A "thesis" is a formula deduced from L1-L3.
Scott listed 68 theses of Łukasiewicz, which included all the axiom systems above, and challenged Wos to find proofs of them from L1-L3 using substitution and detachment, by means of automated deduction.

- That was difficult at the time.
- What constitutes "cheating"? Using any information that depends on knowing a proof already.
- Today this challenge can be met without any cheating.

Using resolution logic for a metalanguage

$P(x)$ means "x is provable". Example: we write

$$P(i(i(x,y),i(i(y,z),i(x,z))))$$

to indicate that L1 is provable. The rule of detachment is axiomatized thus:

$$-P(x) \mid -P(i(x,y)) \mid P(y)$$

Here the vertical bar is disjunction (separating literals of a clause) and $-P(x)$ is the negated literal $P(x)$.

Resolution and condensed detachment

- If one has deduced $P(t)$ and $P(q, r)$ for some terms t, q, and r, then resolution (technically hyperresolution) will try to unify t and q, and if it succeeds with most general unifier σ, it will deduce $P(r\sigma)$.
  Thus hyperresolution steps correspond to condensed detachment at the object level.

Resolution and condensed detachment

- If one has deduced $P(t)$ and $P(q, r)$ for some terms t, q, and r, then resolution (technically hyperresolution) will try to unify t and q, and if it succeeds with most general unifier σ, it will deduce $P(r\sigma)$.
  Thus hyperresolution steps correspond to condensed detachment at the object level.
- However, pure substitution steps can also occur, for example if we try to derive an instance of an axiom. Thus the proofs produced by a theorem prover using this approach will contain condensed detachment steps, and pure substitution steps. This "P-predicate" approach is basic to the application of resolution theorem-proving to particular systems of logic.

Resolution and Detachment

- A resolution inference: From

$$-P(x) \mid -P(i(x,y)) \mid P(y)$$

and $P(i(i(n(x),x),x)$ and $P(i(n(n(x),n(x)))$ infer

$$P(n(x))$$

Resolution and Detachment

- A resolution inference: From

$$-P(x) \mid -P(i(x,y)) \mid P(y)$$

and $P(i(i(n(x),x),x)$ and $P(i(n(n(x),n(x)))$ infer

$$P(n(x))$$

- This corresponds to a detachment inference at the object level: from $i(i(n(x),x),x)$ and $i(n(n(x),n(x))$ infer $n(x)$.

Resolution and Detachment

- A resolution inference: From

$$-P(x) \mid -P(i(x,y)) \mid P(y)$$

  and $P(i(i(n(x),x),x)$ and $P(i(n(n(x),n(x)))$ infer

$$P(n(x))$$

- This corresponds to a detachment inference at the object level: from $i(i(n(x),x),x)$ and $i(n(n(x),n(x))$ infer $n(x)$.
- By searching for resolution proofs at the meta-level we can find detachment-and-substitution proofs at the object level.

Deriving Church from L1-L3

Here is how we express the problem in a file for a theorem-prover. Recall that the goal is to derive $P(i(x, i(y, x))$ and two other formulas. The variables are implicitly universally quantified. Theorem provers always work by searching for a contradiction, so we have to negate this goal, which means that P becomes not P, written $-P$, and the conjunction becomes a disjunction (written with vertical bar). Then we Skolemize, so the existentially quantified variables x, y are replaced by constants a, b. Then the final form of the negated goal is

```
-P(i(a,(i(b,a))))                   % C1
| -P(i(i(a,i(b,c)),i(i(a,b),i(a,c))))  % C2
| -P(i(i(n(a),n(b)),i(b,a))).        % C3
```

Deriving Church from L1-L3

Now we put the negated goal together with the axioms. The prover is supposed to derive a contradiction from the following clauses:

```
P(i(i(x,y),i(i(y,z),i(x,z)))).  % L1
P(i(i(n(x),x),x)).              % L2
P(i(x,(i(n(x),y)))).            % L3
-P(x) | -P(i(x,y)) | P(y).    % condensed detachment
-P(i(a,(i(b,a))))                     % C1
| -P(i(i(a,i(b,c)),i(i(a,b),i(a,c))))  % C2
| -P(i(i(n(a),n(b)),i(b,a))).          % C3
```

Deriving Church from L1-L3

Now we put the negated goal together with the axioms. The prover is supposed to derive a contradiction from the following clauses:

- ```
  P(i(i(x,y),i(i(y,z),i(x,z)))).  % L1
  P(i(i(n(x),x),x)).              % L2
  P(i(x,(i(n(x),y)))).            % L3
  -P(x) | -P(i(x,y)) | P(y).   % condensed detachment
  -P(i(a,(i(b,a))))                           % C1
  | -P(i(i(a,i(b,c)),i(i(a,b),i(a,c))))  % C2
  | -P(i(i(n(a),n(b)),i(b,a))).              % C3
  ```
- Go ahead, try to do it with pencil and paper.
  ```

How hard can it be?

Is it easy or difficult to derive C1-C3 from L1-L3 yourself with pencil and paper?
Remember, a natural-deduction proof is not the same as a detachment-and-substitution proof! We are not claiming that it is an impossible thing to do; after all, Łukasiewicz gave this proof, and similar ones for the other axiom systems mentioned above. We are making the much weaker claim that it is not a trivial thing to do.

A computer-generated proof of Church from Lukas

```
16 [3,3] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
17 [3,4] P(i(i(x,y),i(i(n(x),x),y))).
19 [3,5] P(i(i(i(n(x),y),z),i(x,z))).
20 [5,4] P(i(n(i(i(n(x),x),x)),y)).
24 [16,16] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
29 [16,19] P(i(i(x,n(y)),i(y,i(x,z)))).
31 [19,5] P(i(x,i(n(i(n(x),y)),z))).
32 [19,4] P(i(x,x)).
39 [3,29] P(i(i(i(x,i(y,z)),u),i(i(y,n(x)),u))).
45 [3,24] P(i(i(i(i(x,y),i(z,i(x,u))),v),i(i(z,i(y,u)),v)))
46 [24,17] P(i(i(x,i(n(y),y)),i(i(y,z),i(x,z)))).
```

A computer-generated proof of Church from Lukas

```
59 [3,31] P(i(i(i(n(i(n(x),y)),z),u),i(x,u))).
67 [39,46] P(i(i(n(x),n(y)),i(i(x,z),i(y,z)))).
71 [19,67] P(i(x,i(i(x,y),i(z,y)))).
76 [67,20] P(i(i(i(i(n(x),x),x),y),i(z,y))).
84 [3,71] P(i(i(i(i(x,y),i(z,y)),u),i(x,u))).
93 [84,84] P(i(i(x,y),i(x,i(z,i(u,y))))).
100 [84,19] P(i(n(x),i(x,i(y,z)))).
108 [3,100] P(i(i(i(x,i(y,z)),u),i(n(x),u))).
129 [3,108] P(i(i(i(n(x),y),z),i(i(i(x,i(u,v)),y),z))).
132 [108,46] P(i(n(x),i(i(y,z),i(x,z)))).
141 [3,132] P(i(i(i(i(x,y),i(z,y)),u),i(n(z),u))).
```

A computer-generated proof of Church from Lukas

```
149 [141,59] P(i(n(x),i(y,i(x,z)))).
155 [46,149] P(i(i(i(x,y),z),i(n(x),z))).
183 [93,32] P(i(x,i(y,i(z,x)))).
188 [46,183] P(i(i(i(x,y),z),i(y,z))).
192 [3,183] P(i(i(i(x,i(y,z)),u),i(z,u))).
207 [192,4] P(i(x,i(y,x))).                    %C1
280 [155,188] P(i(n(i(x,y)),i(y,z))).
296 [16,76] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
316 [39,296] P(i(i(n(x),n(y)),i(z,i(y,x)))).
517 [129,4] P(i(i(i(x,i(y,z)),x),x)).
531 [16,517] P(i(i(x,i(x,i(y,z))),i(x,i(y,z)))).
```

A computer-generated proof of Church from Lukas

```
542 [45,531] P(i(i(i(x,y),i(y,z)),i(i(x,y),i(x,z)))).
549 [531,316] P(i(i(n(x),n(y)),i(y,x))).          % C3
580 [188,542] P(i(i(x,y),i(i(z,x),i(z,y)))).
607 [580,580] P(i(i(x,i(y,z)),i(x,i(i(u,y),i(u,z))))).
776 [607,280] P(i(n(i(x,y)),i(i(z,y),i(z,u)))).
796 [4,776] P(i(i(x,i(x,y)),i(x,y))).
808 [580,796] P(i(i(x,i(y,i(y,z))),i(x,i(y,z)))).
880 [45,808] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).  % C2
```

Features of this kind of problem

Here are some things that make this subject amenable to automated deduction:

Features of this kind of problem

Here are some things that make this subject amenable to automated deduction:

- There are few axioms, so we have a chance of not getting swamped by deriving thousands of irrelevant conclusions.

Features of this kind of problem

Here are some things that make this subject amenable to automated deduction:

- There are few axioms, so we have a chance of not getting swamped by deriving thousands of irrelevant conclusions.
- Not much background knowledge is required, so we won't require a huge database of known facts.

Features of this kind of problem

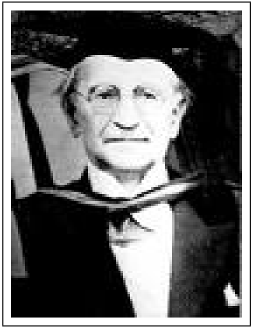
Here are some things that make this subject amenable to automated deduction:

- There are few axioms, so we have a chance of not getting swamped by deriving thousands of irrelevant conclusions.
- Not much background knowledge is required, so we won't require a huge database of known facts.
- Not many variables are required, so we won't get swamped by too many variables (as happens sometimes in elementary geometry, for example)

Jan Łukasiewicz, wedding photo 1929

Jan Łukasiewicz

Łukasiewicz' book was in press in 1939. The press, his home, all his books and all copies of his manuscripts were bombed and burnt. In July, 1944 he left Poland for Germany, in the middle of the Allied invasion, possibly afraid of reprisals against his wife, who (although Polish) was a German sympathizer. He lived out his life in exile in Dublin. Here he is as professor in Dublin.



Meredith's single axiom (1953)

$$i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))$$

Suppose one tries to verify that this is a single axiom "from scratch" by deriving, for example, C1-C3 from it. In 1992 the state of the art was that 236 hours of computer time would get you C3, but not C1 and C2.

Meredith's single axiom (1953)

$$i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))$$

Suppose one tries to verify that this is a single axiom "from scratch" by deriving, for example, C1-C3 from it. In 1992 the state of the art was that 236 hours of computer time would get you C3, but not C1 and C2.

Of course, you can put in as hints the formulas of Meredith's known proof, and get a prover to "find" the known proof. If you still think propositional logic is trivial, try duplicating Meredith's feat, and find a proof. (His proof has 41 steps.)

Meredith's single axiom (2008 update)

In 2008, I suggested to Wos that he try to prove some known axiom system for propositional logic from Meredith's single axiom without "cheating" in any way; that is, without using any information that ultimately derives from Meredith's proof.

Using a recently invented technique, the "subformula strategy", one is able to find a proof of length 183, and one only has to wait 9900 seconds (almost three hours) for the proof to be found. No information about Meredith's proof is used.

This time and proof length can be improved upon, still without using any information about Meredith's proof; one can find a 135 step proof in less than 23 minutes by some slight refinements of the technique. Once any proof at all is in hand, there are well-known techniques to get a shorter proof.

How some theorem-provers work

Create a list called sos (set of support) of clauses, containing all or some of the input.

While list sos is not empty, do the following:

- Select a clause A to be the "given clause", and remove it from sos.
- for each clause B not on sos, if a new clause C can be inferred from A and B, do so.
- Possibly reject C as not fruitful, or mark it as possibly fruitful (to make it be selected sooner)
- If C is a unit clause (only one literal) check for a one-step contradiction ("unit conflict")
- If C is not rejected, add it to sos.

Where artistry comes in

- Select the next given clause
- Grounds for rejection
- Ways to recognize fruitful formulas

Weights

Sometimes the given clause is chosen to have smallest *weight* among clauses on sos. By default, weight is the total number of symbols, but (depending on the prover) you have ways to control that.

One can cause a clause to be rejected by giving it weight more than *max_weight*. One can cause it to be preferred by giving it a low weight.

Using weights to reject unwanted terms

As an example of the use of weights, we can reject all double negations by using $n(n(x)) = junk$, along with $n(junk) = junk$ and $i(x, junk) = junk$ and $(junk, x) = junk$, and then give $junk$ a weight larger than *max_weight*. Any double negation that is deduced will be immediately discarded.

Resonators

We say that two formulas *resonate* if they have the same form, when any variable matches any other variable. Thus $i(x, i(x, x))$ resonates with $i(y, i(x, x))$.

To use a formula B as a *resonator*, we specify that any formula that resonates with B will have a low weight.

As an example of the use of resonators, we could put in all the steps of a known proof (say for example Meredith's 1953 proof) as resonators, specifying a low weight, and then specify that any formulas heavier than that weight should be rejected. This forces the prover to "find" the given proof.

The Subformula Strategy

This strategy consists in using all the subformulas of the goal, or of the axioms, or of some other theorems or axiom systems in the same logic, as resonators. This amazingly simple strategy was not discovered in 1970, 1980, 1900, or 2003, but in 2008.

It is this simple technique that enables automated deduction today to reach the levels of deductive power of Meredith and Łukasiewicz. In particular, this was the technique used to derive Church's 3-base from Meredith's single axiom in three hours, just using the subformulas of the single axiom as resonators.

The improvements mentioned came from using the subformulas of other known axiom systems as resonators as well.

It is worth noting that the change since 1992 is not accounted for by faster computers or larger memory. This could have been done in 1992 if somebody had thought of the subformula strategy then!

Łukasiewicz's single axiom (1930's)

$$(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x)))))$$

Łukasiewicz's 23-symbol single axiom seems to be easier than Meredith's 21-symbol axiom: the same resonators yield a 94-step proof of L1-L3 in less than one minute.
Incidentally, Łukasiewicz never published a proof of a known axiom system from this axiom–the first published proof was found in 1999. Now, when no proof was previously published, you can't exactly have "cheated", but the proof did rely on information from many other deductions. But with the subformula strategy, all you need is the axiom itself.

Many other logical systems can be investigated by these same methods. See Dolph Ulrich's web pages:
Ulrich's home page
and his survey article in the special issue of the Journal of Automated Reasoning, 2001, for more information and references.
We will look at only a few examples.

Intuitionistic Propositional Logic

- More interesting and subtle than classical logic
- Far less obvious that there is a decision procedure.
- Unlike in the classical case, disjunction and conjunction are not definable in terms of implication and negation.
- Still one considers interesting fragments.

Hilbert's 4-base, 1922

$$
\begin{array}{ll}
i(i(p,i(p,q)),i(p,q)) & H1 \\
i(i(q,r),i(i(p,q),i(p,r))) & H2 \\
i(i(p,i(q,r)),i(q,i(p,r))) & H3 \\
i(p,i(q,p)) & H5
\end{array}
$$

Hilbert's 3-base, 1930

$i(i(p, i(p, q)), i(p, q))$	$H1$
$i(i(p, q), i(i(q, r), i(p, r)))$	$H6$
$i(p, i(q, p))$	$H5$

Łukasiewicz's 2-base

$$
\begin{array}{ll}
i(i(p,i(q,r)),i(i(p,q),i(p,r))) & C2 \\
i(p,i(q,p)) & H5
\end{array}
$$

Meredith's 2-base

$$
\begin{array}{lr}
i(i(p,q),i(i(p,i(q,r)),i(p,r))) & M1 \\
i(p,i(q,p)) & H5
\end{array}
$$

Exercise: From each base, derive the others

In July, 2008, I used Otter with the subformula strategy and the "recursive tail strategy" to derive each of these bases from the others. It was straightforward. (I am not claiming any originality here, this may have been done before.)

Meredith's two single axioms

$$i(i(i(p,q),r),i(s,i(i(q,i(r,t)),i(q,t)))) \quad 1953$$
$$i(t,i(i(p,q),i(i(i(s,p),i(q,r)),i(p,r)))) \quad 1963$$

- In 2008, it is easy to check (by machine) that these are single axioms and to derive them from the other bases for this logic.
- This was first done by machine in 2003, but then it was not so easy.
- See Chapter 4 of *Automated Reasoning and the Discovery of Missing and Elegant Proofs*, by Wos and Pieper.

Ulrich has ten more single axioms!

- Dolph (Ted) Ulrich has ten more single axioms of this calculus, five of which he published in 1999, and five he has discovered since then.
- He also has 36 candidates that might or might not be 17-symbol single axioms
- and four 15-symbol candidates, which he conjectures are not single axioms.

See Ulrich's home page at
http://web.ics.purdue.edu/ dulrich/Home-page.htm
Logicians are currently working on these systems, which I copied from the yellowed pages of the Appendix of Prior's *Formal Logic*. (Anyone interested in these things should get a copy of that book.)

Axiomatizing the CN fragment of intuitionistic logic

In addition to the axioms for implication the following system was given by Kolmogorov (1925).

$$i(i(x, n(x)), n(x)))$$
$$i(x, i(n(x), y))$$

Full intuitionistic logic

Using a for disjunction and k for conjunction we have the following system from Horn (1962)

$$
\begin{array}{l}
i(x, i(x, y)) \\
i(i(x, i(y, z)), i(i(x, y), i(x, z))) \\
i(k(x, y), x) \\
i(k(x, y), y) \\
i(i(x, y), i(i(x, z), i(x, k(y, z)))) \\
i(x, a(x, y)) \\
i(y, a(x, y)) \\
i(i(x, z), i(i(y, z), i(a(x, y), z))) \\
i(i(x, n(y)), i(y, n(x))) \\
i(n(x), i(x, y))
\end{array}
$$

A variant of Horn's system

Horn's system is "separable", i.e. each theorem can be proved without using any logical connectives not in the theorem. Other authors have used an axiom system in which, instead of Horn's axiom

$$i(i(x, n(y), i(y, n(x)))$$

we take

$$i(i(x, n(x)), n(x))$$

The other nine axioms are not changed.

A variant of Horn's system

Horn's system is "separable", i.e. each theorem can be proved without using any logical connectives not in the theorem. Other authors have used an axiom system in which, instead of Horn's axiom

$$i(i(x, n(y), i(y, n(x)))$$

we take

$$i(i(x, n(x)), n(x))$$

The other nine axioms are not changed. It takes more than three hours to find a 20-step proof of Horn's axiom. The proof involves conjunction, as well as implication and negation.

A new metatheorem

Wos and I tried to find a proof of Horn's axiom from just the four axioms mentioning only implication and negation. This succeeds in less than a minute.

Conclusion: the variant axiom system is also separable, i.e. each theorem can be proved without using any logical connectives not in the theorem.

This is a new metatheorem (as far as we know).

Heyting's theses

- Heyting's 1930 paper (not his book) lists more than fifty intuitionistically valid propositional formulas.
- Heyting does not provide complete proofs. Moreover, his system of axioms and rules has more rules than modus ponens and substitution.
- In August, 2008, I tried to use Otter to derive these theorems from Horn's axioms. After deleting those theses that are instances of Horn's axioms, there are 51 theses to prove.

Automated deduction of Heyting's theses

- Result: Using the subformula strategy to give preference to subformulas of the goals and axioms, we proved 24 of the 51 formulas, with most of those theorems proved in the first minute.
- By using the steps of those proofs as resonators, and adjusting Otter's parameters, Wos and I proved 36 of the 51.
- Iterating this process, and changing to hints instead of resonators, we so far proved all but two of these theses. This is work in progress.
- Some of the proofs we found are quite long. One is 93 steps; several are of level 15 or more.

Single axioms for Intuitionistic Propositional Calculus

- Tarski (1930) states that any CN system containing $CpCqp$ and $CpCqCCpCqrr$ has a single axiom; the proof may be in Leśniewski (1929).

- Rezus (1982) gave methods to produce such axioms explicitly, but the axioms so produced are long (e.g. 66 symbols for the CN fragment of intuitionistic logic)

- If anyone has produced a shorter single axiom for the CN fragment of intuitionistic logic, I do not know about it.

- Rezus also shows in principle how to construct proofs.

Discarding double negations

- Double negations are formulas of the form $n(n(x))$.
- It is often a useful strategy to discard double negations.
- n(n(x)) = junk.
- But could we be missing proofs this way?

Double Negation Elimination

- A theory T with double-negation-free axioms is said to admit double negation elimination if whenever T proves a theorem without double negations, then it has a proof without double negations.

Double Negation Elimination

- A theory T with double-negation-free axioms is said to admit double negation elimination if whenever T proves a theorem without double negations, then it has a proof without double negations.
- Beeson, Wos, and Veroff proved that all common axiom systems admit double-negation elimination.

Double Negation Elimination

- A theory T with double-negation-free axioms is said to admit double negation elimination if whenever T proves a theorem without double negations, then it has a proof without double negations.
- Beeson, Wos, and Veroff proved that all common axiom systems admit double-negation elimination.
- Classical logic, intuitionistic logic, and multi-valued logic.

Role of computers in the proof of double-negation elimination

- We gave general conditions on a theory T that should be satisfied.
- Those involved the provability in T of certain axioms.
- We used a theorem-prover to prove lemmas of the form, a given theory T proves a given theorem by condensed detachment.
- We published in *Studia Logica* on the logical merits of the result, and half the proofs were computer-generated.

The pushback lemma

- A key lemma says that a proof by detachment and substitution from axioms T can be converted to a proof by modus ponens only from substitution instances of the axioms.
- The substitutions are “pushed back” to the beginning of the proof.

An example

$$i(i(n(x), n(i(i(n(y), n(z)), n(z)))),$$
$$n(i(i(n(i(n(x), y)), n(i(n(x), z))), n(i(n(x), z)))))$$

This formula is provable in many-valued propositional logic, and is itself double-negation free.
(Why it is interesting is not relevant to our story.)
By our theorem it should have a double-negation free proof, but Wos had been unable to find one.

The 200 kilobyte proof

- Wos provided a proof of 45 condensed-detachment steps, 16 of whose lines involved a double negation.

The 200 kilobyte proof

- Wos provided a proof of 45 condensed-detachment steps, 16 of whose lines involved a double negation.
- Beeson used this proof as input to a computer program implementing the algorithms implicit in the proof of our elimination theorem.

The 200 kilobyte proof

- Wos provided a proof of 45 condensed-detachment steps, 16 of whose lines involved a double negation.
- Beeson used this proof as input to a computer program implementing the algorithms implicit in the proof of our elimination theorem.
- Output: a double-negation-free proof by modus ponens of the example, from substitution instances of A1– A4.

The 200 kilobyte proof

- Wos provided a proof of 45 condensed-detachment steps, 16 of whose lines involved a double negation.
- Beeson used this proof as input to a computer program implementing the algorithms implicit in the proof of our elimination theorem.
- Output: a double-negation-free proof by modus ponens of the example, from substitution instances of A1– A4.
- The proof was 796 lines, and many of its lines involved thousands of symbols. The input proof takes about 3.5 kilobytes, the output proof about 200 kilobytes.
- Now we know what the “condensed” means in “condensed detachment”!

The 200 kilobyte proof

- Wos provided a proof of 45 condensed-detachment steps, 16 of whose lines involved a double negation.
- Beeson used this proof as input to a computer program implementing the algorithms implicit in the proof of our elimination theorem.
- Output: a double-negation-free proof by modus ponens of the example, from substitution instances of A1– A4.
- The proof was 796 lines, and many of its lines involved thousands of symbols. The input proof takes about 3.5 kilobytes, the output proof about 200 kilobytes.
- Now we know what the "condensed" means in "condensed detachment"!
- Proof was shortened by McCune and Wos to 37 steps.

Semantics of many-valued logic

Truth values are real numbers in $[0,1]$. Negation and implication are interpreted by the following functions from $[0,1]$ to $[0,1]$:

$$\begin{aligned} c(x,y) &= \min(1, 1-x+y) \\ n(x) &= 1-x \end{aligned}$$

Writing $[\![p]\!]$ for the truth value of p, we have by definition

$$\begin{aligned} [\![n(p)]\!] &= n([\![p]\!]) \\ [\![i(p,q)]\!] &= c([\![p]\!], [\![q]\!]) \end{aligned}$$

Łukasiewicz's A1-A5

Lukasiewicz defined the many-valued sentential calculus $L_{\aleph_0}$ and gave the following axioms.

$$
\begin{array}{ll}
i(x, i(y, x)) & A1 \\
i(i(x, y), i(i(y, z), i(x, z))) & A2 \\
i(i(i(x, y), y), i(i(y, x), x) & A3 \\
i((i(x, y), i(y, x)), i(y, x)) & A4 \\
i(i(n(x), n(y)), i(y, x)) & A5
\end{array}
$$

Completeness theorem

- A subset I of $[0,1]$ closed under c and n can serve as a set of truth values for MV.
- At one extreme we can take $I = \{0,1\}$, recovering two-valued logic, and at the other extreme we can take $I = [0,1]$.
- Fix an infinite set I of truth values. A sentence of MV is defined to be valid for I if its truth value is always 1, regardless of the truth values assigned to its variables.
- The completeness theorem is that ϕ is derivable from $A1 - A5$ if and only if ϕ is valid for I.

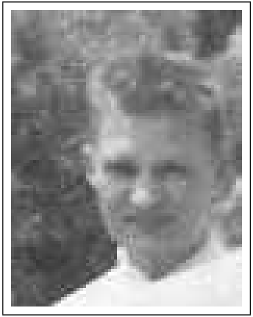
This result was conjectured by Łukasiewicz and later proved by his student Wajsberg, according to Tarski; but Wajsberg never published his proof.

Mordechai Wajsberg

Wajsberg was fortunate to be a student of Łukasiewicz , but had the misfortune to be a Polish Jew at the wrong point in history, and he perished in the Holocaust. He was last seen on the train to Treblinka (a Nazi death camp) in 1943.

Rose and Rosser

The first published proof of the completeness theorem is by Rose and Rosser (1958).



J. Barkley Rosser (1950)

Dependence of A4

The fact that A4 can be derived from A1-A3 and A5 was proved independently by Meredith and Chang. Their proofs appeared on adjacent pages in the same journal in 1958.
In 1992 this proof was too hard to find automatically (without cheating). McCune and Wos reported failure at CADE-11 that summer (although they reported many successes as well).

Success using the tail strategy

- Shortly after the failure reported in 1992, Wos succeeded in finding a no-cheating proof of the dependence of A4.
- The method was the "recursive tail strategy", which consists in favoring formulas $i(x, y)$ with short "tails" y.
- This is done by counting the "head" x double when computing the weight.
- A good strategy helps more than Moore's law

Disjunction and Conjunction in MV logic

Multi-valued logic, like its cousin linear logic, has two disjunctions A and B and two conjunctions K and L, definable in different ways in terms of implication and negation. Rose and Rosser say ([**?**], pp. 11–12):

> *With both A and B serving as disjunctions and both K and L serving as conjunctions, one can write a number of possible distributive laws. Some are not valid, and of the valid ones we have been able to prove only two from the axiom schemes A1-A4.*

Distributive laws in MV logic

There were two valid distributive laws that Rose and Rosser could not prove. One of them was

$$CKpAqrAKpqKpr$$

It is a simple exercise to show that this formula is semantically valid.
When translated into the CN fragment, this distributive law becomes quite a long formula:

$$CNCCNpNCCqrrNCCqrrCCNCCNpNqNqNCCNpNrNrNCCN$$

Fitelson and Harris

- Fitelson and Harris used Otter to find a condensed-detachment proof of this distributive law from A1-A3 and A5.

Fitelson and Harris

- Fitelson and Harris used Otter to find a condensed-detachment proof of this distributive law from A1-A3 and A5.
- Their method involved equality between formulas as well as condensed detachment.

Fitelson and Harris

- Fitelson and Harris used Otter to find a condensed-detachment proof of this distributive law from A1-A3 and A5.
- Their method involved equality between formulas as well as condensed detachment.
- This technique was invented by Wos and McCune.

Fitelson and Harris

- Fitelson and Harris used Otter to find a condensed-detachment proof of this distributive law from A1-A3 and A5.
- Their method involved equality between formulas as well as condensed detachment.
- This technique was invented by Wos and McCune.
- Veroff helped convert this bidirectional proof into a forward proof with equality.

Fitelson and Harris

- Fitelson and Harris used Otter to find a condensed-detachment proof of this distributive law from A1-A3 and A5.
- Their method involved equality between formulas as well as condensed detachment.
- This technique was invented by Wos and McCune.
- Veroff helped convert this bidirectional proof into a forward proof with equality.
- McCune has an algorithm to convert proofs with equality to condensed detachment.

Fitelson and Harris, continued

- Result: a condensed-detachment proof, almost 100 steps long.

Branden Fitelson

Fitelson and Harris, continued

- Result: a condensed-detachment proof, almost 100 steps long.
- Later shortened by Wos by discarding double negations etc. to 85 steps

Branden Fitelson

Fitelson and Harris, continued

- Result: a condensed-detachment proof, almost 100 steps long.
- Later shortened by Wos by discarding double negations etc. to 85 steps
- Fitelson and Harris published the 85-step proof, the world's first.

Branden Fitelson

Harris's even-shorter proof

That, however, is not the end of the story. Harris decided to *think* about the problem, not in the CN fragment, but in the original language.

- Harris found a substitution and detachment proof by hand.
- Using resonance and hints based on Harris's hand-constructed proof, Fitelson and Harris found a 61-step proof, which they say they find "more intuitive and explanatory than the CN fragment proof."
- Although this time, the machine proof came first, the human proof is allegedly better.
- But they they never tried (or never reported on) using automated deduction in the full language including K and A, which was what Harris used for his hand proof.

Modal Logic

Modal logic is a term for a class of systems formed from one or another sentential logic (or first-order logic for that matter) by adding the "necessity operator" $\Box$. All such systems have the formula formation rule that if A is a formula, so is $\Box A$, and the inference rule, from A infer $\Box A$.

Resolution-based metatheory

To use a resolution language as the metalanguage for modal logic, we simply write `l(A)` for $\Box A$, and the inference rule becomes

```
-P(x) | P(l(x)).
```

This rule is known as RN, the rule of necessitation.

Various axiomatizations

In modal logic, there is usually a "possibility operator" $\Diamond$, which in most logics can be defined by

$$\Diamond A := \neg\Box\neg A.$$

Two modal axioms of interest are

$$\begin{array}{ll} \Box(p \supset q) \supset (\Box p \supset \Box q) & (K) \\ \Box p \supset p & (T) \end{array}$$

The theory CKT (classical modal sentential logic) consists of axioms K and T, the rule RN, and some base for classical logic; for definiteness we take L1-L3.

Triviality in classical modal logic

A modal logic including T is called "trivial" if it proves $A \supset \Box A$. The following are examples of principles that, when added to CKT, produce a trivial logic:

$$p \supset \Diamond \Box p \qquad (W)$$
$$\Box(\Diamond p \supset \Diamond q) \supset \Box(p \supset q) \qquad (F)$$

These two triviality results are due to Williamson and Fine, respectively. Fine also proved a triviality result for the theory CKF, which does not include T. Namely, CKF proves

$$\Box(p \leftrightarrow \Diamond p)$$

Triviality in non-classical modal logics

Fitelson investigated whether, in this result of Fine, one can weaken classical logic. (Fine's proof definitely uses classical logic.) Using Otter, he was able to show that Fine's triviality result still holds if CKF is replaced by XKF, where X is either intuitionistic logic, or three-valued logic (either Łukasiewicz's three-valued logic or Kleene's).

Triviality in non-classical modal logics

Examining the proofs, Fitelson found the following four axioms:

$$i(i(p,q),i(i(r,p),i(r,q)))$$
$$i(i(p,q),i(n(q),n(p)))$$
$$i(i(i(p,q),r),i(i(q,p),r))$$
$$i(i(i(p,q),r),i(q,r))$$

These four axioms suffice to prove $\Box(p \leftrightarrow \Diamond p)$, or as it would be expressed formally, to prove both $l(i(p,n(l(n(p)))))$ and $l(i(n(l(n(p))),p))$.

In other words, the triviality theorem works for XKF, where X is the above four axioms; and that implies all the other generalizations of Fine's result.

These are new results, found with the aid of computers.

Areas we don't have time to discuss

- Combinatory logic(s) (fixed point properties)
- Combinatory logic as a first-order way to do lambda calculus
- Combinatory logic as a way to define the quantifiers, allowing us to formalize the metatheory of first-order logic.
- Equivalential calculus (another invention of Łukasiewicz)
- Relevance logics
- Provability logics
- Use of model-finding programs for unprovability results
- Use of quantifier-elimination in the reals to formalize semantics

Open problems involving proofs in sentential calculi

- There was a list of open problems in Ulrich's 2001 article, and an update about progress on those problems since then can be found on his website. Many of these have to do with single axioms for various theories; how short these can or cannot be, and whether various specific formulas are or are not single axioms.
- There has been a lot of work with shortening the length of proofs; we could also pay more attention to the number of variables and the size (total number of symbols).
- More work needs to be done on equality reasoning and its relation to sentential calculi.

More open problems in sentential calculi

- Can we work with systems containing four or more connectives?
- Can we work with systems not based entirely on condensed detachment? For example, sequent calculi, tableaux calculi, natural deduction calculi.
- Negative results are often harder than positive ones: for example, is there a finite axiomatization of the 2-variable fragment of intuitionistic propositional calculus? (asked by MacKay).

Open problems in formalized metamathematics

- Proving metatheorems formally. There is no problem doing mathematical induction in first-order systems if we specify the instances of induction that are required, e.g. to prove the deduction theorem or the soundness of the double-negation interpretation or, in the future, more complicated interpretations.
- Formalized semantics. More can be done using qepcad. Formalized Kripke semantics?
- What is the exact relationship between Gentzen's cut-elimination theorem and Knuth-Bendix completion?

A research program

- Connect the work in automated deduction to the work of the proof-checking community.

A research program

- Connect the work in automated deduction to the work of the proof-checking community.
- When formalizing a proof (to verify its correctness), a theorem-prover should fill in the small steps.

A research program

- Connect the work in automated deduction to the work of the proof-checking community.
- When formalizing a proof (to verify its correctness), a theorem-prover should fill in the small steps.
- As a first project: Formalize the metatheory of various sentential logics in Mizar, and write a program that translates Otter output (of a proof that some formula A is provable in a logic T) into a Mizar-checkable proof that A is provable in T.

Summary

- The method: formalize the metatheory of some logic using clauses, and then use a theorem-prover to find proofs in that logic.
- You may need to invent new strategies for discarding or favoring certain formulas.
- Strategy is more important than raw computer power.
- With today's computers and today's strategies, this method is as good as the great logicians of the twentieth century at finding proofs.
- Some open problems have been settled, and some past results improved.
- There is a lot of room for future work in this area.