Defining the semantics of proof evidence

Dale Miller

Inria Saclay & LIX, École Polytechnique
Palaiseau, France

7 August 2015, HaPoC Session, CLMPS 2015, Helsinki

Joint work with Roberto Blanco, Zakaria Chihani, Quentin Heath, Danko Ilik, Tomer Libal, Fabien Renaud, Giselle Reis

For more, see papers in: CADE 2013, CPP 2011/13/15.

Outline

- Formal proofs in the modern world.
- A proposal for separating formal proofs from provenance.
- Outline how modern proof theory research can provide a framework for defining a wide range of proof evidence.

Some roles for formal proofs in Mathematics

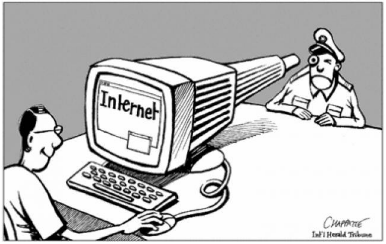
- Frege, Hilbert, Church, Gödel, etc used Frege/Hilbert (formal) proofs to increase trust in foundational issues.
- Voevodsky uses Coq to reduce abstract proofs to computation in order to survive possible inconsisencies in mathematics.
- Hales and Gonthier have use modern theorem provers (Isabelle, Coq, and HOL) to formally prove the Four color theorem, the Feit–Thompson (odd-order) theorem, and the Kepler conjecture.

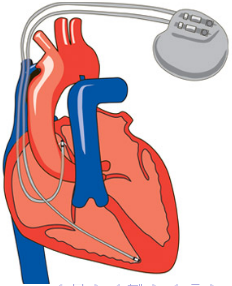
Some roles for formal proofs in Mathematics

- Frege, Hilbert, Church, Gödel, etc used Frege/Hilbert (formal) proofs to increase trust in foundational issues.
- Voevodsky uses Coq to reduce abstract proofs to computation in order to survive possible inconsisencies in mathematics.
- Hales and Gonthier have use modern theorem provers (Isabelle, Coq, and HOL) to formally prove the Four color theorem, the Feit–Thompson (odd-order) theorem, and the Kepler conjecture.

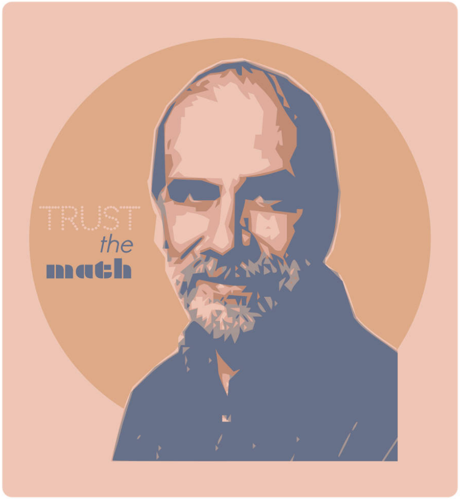
There are several places in the modern, digital world where formal proofs can be used.

What can we trust?





In cryptology: Trust the math



Bruce Schneier

In software correctness: Trust the proof!

With software systems, there are many things to trust.

- verification condition generators
- type checkers, type inference, abstract interpretation
- compilers
- printers and parsers
- theorem provers

All this is overwhelming. A modest goal:

Provide the framework so that we can at least trust proofs.

We restriction our of attention to *formal proofs*, generated and checked by computer tools.

The current situation with formal proofs

Most proof production and checking is technology based.

If you change the version number of a prover, it may not recognized its earlier proofs.

Most proofs are locked into the technology.

Some bridges are now being built between different provers, but these are affected by two version numbers.

The current situation with formal proofs

Most proof production and checking is technology based.

If you change the version number of a prover, it may not recognized its earlier proofs.

Most proofs are locked into the technology.

Some bridges are now being built between different provers, but these are affected by two version numbers.

A recent panel discussion (PxTP 2015, 2 August) revealed that practitioners do not alway trust their theorem provers. They use other provers to double check their work.

The vision: The network *is* the prover

Goal: Permit the formal methods community to become a network of communicating provers.

Proof certificates: documents that circulate and denote proofs.

Approach: Provide formal definitions of “proof evidence” so that proof certificates can be checked by *trusted checkers*.

But: There is a wide range of “proof evidence.”

- proof scripts for steering a theorem prover to a proof
- resolution refutations, natural deduction, tableaux, etc
- winning strategies, simulations

Outline

- Formal proofs in the modern world.
- A proposal for separating formal proofs from provenance.
- Outline how modern proof theory research can provide a framework for defining a wide range of proof evidence.

The need for frameworks

Three central questions:

- How can we manage so many "proof languages"?
- Will we need just as many proof checkers?
- How does this improve trust?

Computer scientists have seen this kind of problem before.

The need for frameworks

Three central questions:

- How can we manage so many "proof languages"?
- Will we need just as many proof checkers?
- How does this improve trust?

Computer scientists have seen this kind of problem before.

We develop *frameworks* to address such questions.

- lexical analysis: finite state machines / transducers
- language syntax: grammars, parsers, attribute grammars, parser generators
- programming languages: denotational and operational semantics

A non-goal: didactic aspects of formal proofs

We do not assume that humans will necessarily be able to read or learn from formal proofs.

Consider formal proofs of the following kind of theorems.

- 2147483647 is prime.
- A certain program will not produce a buffer overflow error.
- There is no path between two points in some reachability graph.

Of course, having tools to *browse* and *interact* with a formal proof is certainly desirable. Eventually.

Earliest notion of formal proof

Frege, Hilbert, Church, Gödel, etc, made extensive use of the following notion of proof:

> *A proof is a list of formulas, each one of which is either an* axiom *or the conclusion of an* inference rule *whose premises come earlier in the list.*

While granting us trust, there is little useful structure here.

The first programmable proof checker

Copyrighted Material

Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

78

Michael J. Gordon
Arthur J. Milner
Christopher P. Wadsworth

Edinburgh LCF

Springer-Verlag
Berlin Heidelberg New York

Copyrighted Material

LCF/ML (1979) viewed proofs as slight generalizations of such lists.

ML provided types, abstract datatypes, and higher-order programming in order to increase confidence in proof checking.

Many provers today (HOL, Coq, Isabelle) follow LCF principles.

Outline

- Formal proofs in the modern world.
- A proposal for separating formal proofs from provenance.
- Outline how modern proof theory research can provide a framework for defining a wide range of proof evidence.

More recent advances: Atoms and molecules of inference

Atoms of inference

- Gentzen's **sequent calculus** first provided these: introduction, identity, and structural rules
- Girard's **linear logic** refinement of these inference rules
- To account for first-order structure, we also need **fixed points** and **equality**. (eg. McDowell, Tiu, Baelde, et al).

Rules of Chemistry

- **Focused proof systems** show us that some atoms stick together while other atoms form boundaries.

Molecules of inference

- Collections of atomic inference rules that stick together form synthetic inference rules.

Features enabled for proof certificates

- Simple checkers can be implemented.
  Only the atoms of inference and the rules of chemistry (both small and closed sets) need to be implemented in a checker of certificates.

- Certificates support a wide range of proof systems.
  The molecules of inference can be engineered into a wide range of inference rules.

- Certificates are based (ultimately) on proof theory.
  Immediate by design.

- Proof details can be elided.
  Search using atoms will match search in the space of molecules: that is, the checker will not invent new molecules.

Clerks and experts: the office workflow analogy

Imagine an accounting office that needs to check if a certain mound of financial documents (provided by a **client**) represents a legal tax transaction (as judged by the **kernel**).

Experts look into the mound and extract information and

- *decide* which transactions to dig into and
- *release* their findings for storage and later reconsideration.

Clerks take information released by the experts and perform some computations on them, including their *indexing* and *storing*.

Focused proofs alternate between two phases: *positive* (experts are active) and *negative* (clerks are active).

The terms *decide*, *store*, and *release* come from proof theory.

A proof certificate format defines workflow and the duties of the clerks and experts.

Proof checking and proof reconstruction

Clearly, (determinate) computation is built into this paradigm: the clerks can perform such computation.

Proof *reconstruction* might be needed when invoking not-so-expert experts (or ambiguous tax forms).

Non-deterministic computation is part of the mix: non-determinism is an important resource that is useful for proof-compression.

The *LKneg* proof system

Use invertible rules where possible. In propositional classical logic, both conjunction and disjunction can be given invertible rules.

$$\frac{\vdash \cdot; B}{\vdash B}\ \mathit{start} \qquad \frac{\vdash \Delta, L; \Gamma}{\vdash \Delta; L, \Gamma}\ \mathit{store} \qquad \frac{}{\vdash \Delta, A, \neg A; \cdot}\ \mathit{init}$$

$$\frac{\vdash \Delta; \Gamma}{\vdash \Delta; \mathit{false}, \Gamma} \qquad \frac{\vdash \Delta; B, C, \Gamma}{\vdash \Delta; B \vee C, \Gamma} \qquad \frac{}{\vdash \Delta; \mathit{true}, \Gamma} \qquad \frac{\vdash \Delta; B, \Gamma \quad \vdash \Delta; C, \Gamma}{\vdash \Delta; B \wedge C, \Gamma}$$

Here, A is an atom, L a literal, Δ a multiset of literals, and Γ a list of formulas. Sequents have two *zones*.

This proof system provides a decision procedure (resembling conjunctive normal forms).

A small (constant sized) certificate is possible.

The *LKneg* proof system

Use invertible rules where possible. In propositional classical logic, both conjunction and disjunction can be given invertible rules.

$$\frac{\vdash \cdot; B}{\vdash B}\ \mathit{start} \qquad \frac{\vdash \Delta, L; \Gamma}{\vdash \Delta; L, \Gamma}\ \mathit{store} \qquad \frac{}{\vdash \Delta, A, \neg A; \cdot}\ \mathit{init}$$

$$\frac{\vdash \Delta; \Gamma}{\vdash \Delta; \mathit{false}, \Gamma} \quad \frac{\vdash \Delta; B, C, \Gamma}{\vdash \Delta; B \vee C, \Gamma} \quad \frac{}{\vdash \Delta; \mathit{true}, \Gamma} \quad \frac{\vdash \Delta; B, \Gamma \quad \vdash \Delta; C, \Gamma}{\vdash \Delta; B \wedge C, \Gamma}$$

Here, A is an atom, L a literal, Δ a multiset of literals, and Γ a list of formulas. Sequents have two *zones*.

This proof system provides a decision procedure (resembling conjunctive normal forms).

A small (constant sized) certificate is possible.

Consider proving $(p \vee C) \vee \neg p$ for large C.

The *LKpos* proof system

Non-invertible rules are used here.

$$\frac{\vdash B; \cdot; B}{\vdash B}\ \mathit{start} \qquad \frac{\vdash B; \mathcal{N}, \neg A; B}{\vdash B; \mathcal{N}; \neg A}\ \mathit{restart} \qquad \frac{}{\vdash B; \mathcal{N}, \neg A; A}\ \mathit{init}$$

$$\frac{\vdash B; \mathcal{N}; B_i}{\vdash B; \mathcal{N}; B_1 \vee B_2} \qquad \frac{}{\vdash B; \mathcal{N}; \mathit{true}} \qquad \frac{\vdash B; \mathcal{N}; B_1 \quad \vdash B; \mathcal{N}; B_2}{\vdash B; \mathcal{N}; B_1 \wedge B_2}$$

Here, A is an atom and $\mathcal{N}$ is a multiset of negated atoms.
Sequents have three *zones*.

The $\vee$ rule *consumes* some external information or some non-determinism.

An *oracle string*, a series of bits used to indicate whether to go left or right, can be a proof certificate.

A proof in $LKpos$

Let C have several alternations of conjunction and disjunction.

Let $B = (p \vee C) \vee \neg p$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash B; \neg p; p}\ \mathit{init}}{\vdash B; \neg p; p \vee C}\ {\color{blue}*}}{\vdash B; \neg p; (p \vee C) \vee \neg p}\ {\color{blue}*}}{\vdash B; \cdot\ ; \neg p}\ \mathit{restart}}{\vdash B; \cdot\ ; (p \vee C) \vee \neg p}\ {\color{blue}*}}{\vdash B}\ \mathit{start}
$$

The subformula C is avoided. Clever choices $*$ are injected at these points: right, left, left. We have a small certificate and small checking time. In general, these certificates may grow large.

Combining the *LKneg* and *LKpos* proof systems

Introduce two versions of conjunction, disjunction, and their units.

$$t^-, t^+, f^-, f^+, \vee^-, \vee^+, \wedge^-, \wedge^+$$

The inference rules for negative connectives are invertible.

These polarized connectives also exist in linear logic.

Introduce the two kinds of sequent, namely,
$\vdash \Theta \Uparrow \Gamma$: for invertible (negative) rules (Γ a list of formulas)
$\vdash \Theta \Downarrow B$: for non-invertible (positive) rules (B a formula)

LKF : a focused proof systems for classical logic

$$\frac{}{\vdash \Theta \Uparrow \Gamma, t^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, B \qquad \vdash \Theta \Uparrow \Gamma, B'}{\vdash \Theta \Uparrow \Gamma, B \wedge^- B'} \qquad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, f^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, B, B'}{\vdash \Theta \Uparrow \Gamma, B \vee^- B'}$$

$$\frac{}{\vdash \Theta \Downarrow t^+} \qquad \frac{\vdash \Theta \Downarrow B \qquad \vdash \Theta \Downarrow B'}{\vdash \Theta \Downarrow B \wedge^+ B'} \qquad \frac{\vdash \Theta \Downarrow B_i}{\vdash \Theta \Downarrow B_1 \vee^+ B_2}$$

$$\overset{\text{Init}}{\frac{}{\vdash \neg A, \Theta \Downarrow A}} \qquad \overset{\text{Store}}{\frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, C}} \qquad \overset{\text{Release}}{\frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}} \qquad \overset{\text{Decide}}{\frac{\vdash P, \Theta \Downarrow P}{\vdash P, \Theta \Uparrow \cdot}}$$

P is a positive formula; N is a negative formula;
A is an atom; C positive formula or negative literal

Results about LKF

Let B be a propositional logic formula and let $\hat{B}$ result from B by placing $+$ or $-$ on t, f, $\wedge$, and $\vee$ (there are exponentially many such placements).

Theorem. [Liang & M, TCS 2009]

- If B is a tautology then every polarization $\hat{B}$ has an LKF proof.
- If some polarization $\hat{B}$ has an LKF proof, then B is a tautology.

The different polarizations do not change *provability* but can radically change the *proofs*.

Also:

- Negative (non-atomic) formulas are treated linearly (never weakened nor contracted).
- Only positive formulas are contracted (in the Decide rule).

Example: deciding on a simple clause

Assume that Θ contains the formula $a \wedge^+ b \wedge^+ \neg c$ and that we have a derivation that Decides on this formula.

$$\dfrac{\dfrac{\dfrac{}{\vdash \Theta \Downarrow a}\ \mathit{Init} \qquad \dfrac{}{\vdash \Theta \Downarrow b}\ \mathit{Init} \qquad \dfrac{\dfrac{\vdash \Theta, \neg c \Uparrow \cdot}{\vdash \Theta \Uparrow \neg c}\ \mathit{Store}}{\vdash \Theta \Downarrow \neg c}\ \mathit{Release}}{\vdash \Theta \Downarrow a \wedge^+ b \wedge^+ \neg c}\ \wedge^+}{\vdash \Theta \Uparrow \cdot}\ \mathit{Decide}$$

This derivation is possible iff Θ is of the form $\neg a, \neg b, \Theta'$. Thus, the "macro-rule" is

$$\dfrac{\vdash \neg a, \neg b, \neg c, \Theta' \Uparrow \cdot}{\vdash \neg a, \neg b, \Theta' \Uparrow \cdot}$$

Example: Resolution as a proof certificate

- A *clause:* $\forall x_1 \ldots \forall x_n [L_1 \vee \cdots \vee L_m]$
- C_3 is a *resolution* of C_1 and C_2 if we chose the mgu of two complementary literals, one from each of C_1 and C_2, etc.
- If C_3 is a resolvent of C_1 and C_2 then $\vdash \neg C_1, \neg C_2 \Uparrow C_3$ has a short proof (decide depth 2 or less).

Translate a refutation of $C_1, \ldots, C_n$ into a (focused) sequent proof with small holes:

$$\dfrac{\dfrac{\Xi}{\vdash \neg C_1, \neg C_2 \Uparrow C_{n+1}} \qquad \dfrac{\dfrac{\vdots}{\vdash \neg C_1, \ldots, \neg C_n, \neg C_{n+1} \Uparrow \cdot}}{\vdash \neg C_1, \ldots, \neg C_n \Uparrow \neg C_{n+1}}\, \mathit{Store}}{\vdash \neg C_1, \ldots, \neg C_n \Uparrow \cdot}\, \mathit{Cut}$$

Here, Ξ can be replaced with a "hole" bounded by depth 2.

The ProofCert project: recent results

- The FPC framework for first-order (classical and intuitionistic) logics.
- Defined various proof certificate formats:
  - Classical: resolution, expansion trees, matings, CNF, etc.
  - Intuitionistic: natural deduction, various typed λ-calculus.
  - Also: Frege systems, equality reasoning, etc.
- Implemented a reference kernel (using λProlog / Teyjus)
- The intuitionistic checker can “host” the classical kernel, so only one kernel is needed.

The ProofCert project: next steps

Address induction, co-induction, and model checking

Develop certificates for various modal and temporal logics

Treat parallelism in proof structures (using multi-focusing and multi-cut rules)

Develop an approach to theories: set theories, type theories, etc

Design of libraries of theorems and proofs

Related Work

PCC - proof carrying code

TPTP - a library of theorems and proofs, promotes interchange between theorem provers

LF - Logical Framework (dependently typed λ-calclulus)

Dedukti - a proof checker based on dependent typed λ-calculus and functional computations

PVS and "little engines of proof"

Thank you

What relations are there between LF and FPC?

LF: The logical framework of Harper, Honsell, and Plotkin [1987, 1993] (a.k.a. $\lambda\Pi$).

It seems straightforward to encode LF, LFSC (LF with side conditions), and LF modulo (Dedukti) as FPCs.

Alone LF does not seem to have the right "atoms of inference."

- Canonical normal forms provide only one structuring of proofs.
- These lack an analytic notion of classical reasoning and sharing.
- Also lacking is a natural treatment of parallel proof steps.