A foundational approach to proof certificates

An application of proof theory to computer science

Dale Miller

INRIA-Saclay & LIX, École Polytechnique

Freie Universität Berlin
22 February 2013

Can we standardize, communicate, and trust formal proofs?

Funding from ProofCert, an ERC Advanced Grant

Outline

Proof-as-message; proof-as-certificate

A proof can serve the *didactic* purpose of explaining the "why" behind a theorem. A proof has a *message*.

We shall not consider further the role of **proof-as-message** here.

A proof can serve as a *certificate* that a formula is, in fact, true.

Premise for this talk: Both *structural proof theory* and *computer automation* have matured sufficiently to provide a flexible and universal approach to **proof-as-certificate**.

Proof as certification

Proofs are *documents* that are used to *communicate trust* within a *community of agents*.

Agents can be machines and humans. Communities can be spread across time and space.

Societies of agents: both humans and machines

- A *sole mathematician* writes an argument that convinces herself and she then moves to address new problems.
- A collection of *mathematician colleagues* searches for beautiful and deep mathematical concepts.
- An *author of a mathematics text and his readers* is a community in which communications is generally *one-way*.
- *Programmers*, who write code for an OS, and *users* of that OS have a goal of producing useful and secure computer systems.
- A *group of programmers, users, mobile computers, and servers* can form a society that exchanges money for various services (eg, email, news, backups, and cloud computing).

Informal proofs

Informal proofs are readable by humans and are (usually) *didactic*.

We also expect that they lack some details and that they may have errors.

Informal proofs are circulated within societies of humans where they can be evaluated in a number of ways:

- Is the proof proving something interesting?
- Are the assumptions the right ones?
- Are the proof methods appropriate?
- Is this situation an example or a counterexample?

Making an informal proof more formal allows it to communicate across greater distances (in space and time).

Our project starts here: Formal proofs

Our focus:
computer agents communicating and checking proofs

A *formal proof* is a document with a precise syntax that is machine generated and machine checkable.

We do not assume that formal proofs are human-readable.

In principle, an algorithm should make it possible to "perform" the proof described in the document.

Trusted computer tools are used to check proofs so that other human or machine agents come to trust the truth of a formula.

Provers: computer agents that produce proofs

There is a wide range of provers.

- automated and interactive theorem provers
- computer algebra systems
- model checkers, SAT solvers
- type inference, static analysis
- testers

There is a wide range of "evidence" of proof.

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

It is the exception when one prover's evidence is shared with or trusted by another prover.

Require provers to publish their proofs

Since provers do not currently communicate proofs, the trend is to unifying various theorem proving activities into existing frameworks, eg, Isabelle or Coq.

Separate proofs from provenance: insist that provers output their proofs so others can check them.

We shall use the term “proof certificate” for those documents denoting proofs that are circulated between provers.

Goal: A sea change is needed in formal methods

Sun Microsystems (1984): The network *is* the computer

The formal methods community uses many isolated provers technologies: proof assistants (Coq, Isabelle, HOL, PVS, etc), model checkers, SAT solvers, etc.

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

Outline

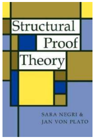
D1: A simple checker can, in principle, check if a proof certificate denotes a proof.

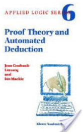
D2: The proof certificate format supports a broad spectrum of proof systems.

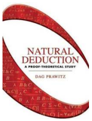
These two desiderata enable the creation of both **marketplaces** and **libraries** of proofs.

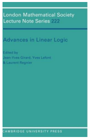
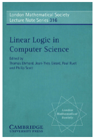
D3: A proof certificate is intended to denote a proof in the sense of structural proof theory.

Structural proof theory is a mature field that deals with deep aspects of proofs and their properties.








London Mathematical Society Lecture Note Series 222
Advances in Linear Logic
CAMBRIDGE UNIVERSITY PRESS

London Mathematical Society Lecture Note Series 316
Linear Logic in Computer Science
CAMBRIDGE UNIVERSITY PRESS

For example: given certificates for

$$\forall x (A(x) \supset \exists y\ B(x,y)) \quad \text{and} \quad A(10),$$

can we extract from them a witness t such that $B(10, t)$ holds?

D4: A proof certificate can simply leave out details of the intended proof.

Formal proofs are often huge. All means to reduce their size need to be available.

- Allow abstractions and lemma.
- Separate *computation* from *deduction* and leave computation traces out of the certificate.
- Permit holes in proofs: we now have a trade-offs between *proof size* and *proof reconstruction* via (bounded) proof search.

Proof checking may involve significant computation in order to *reconstruct* missing proof details.

Which logic?

First-order or higher-order?

Which logic?

First-order or higher-order? Both!

Higher-order (à la Church 1940) seems a good choice since it includes propositional and first-order.

Which logic?

First-order or higher-order? Both!

Higher-order (à la Church 1940) seems a good choice since it includes propositional and first-order.

Classical or intuitionistic logic?

Which logic?

First-order or higher-order? Both!

Higher-order (à la Church 1940) seems a good choice since it includes propositional and first-order.

Classical or intuitionistic logic? Both!

Imagine that these two logics fit together in one larger logic. Following Gentzen (LK/LJ), Girard (LU), Liang & M (LKU, PIL).

Which logic?

First-order or higher-order? Both!

Higher-order (à la Church 1940) seems a good choice since it includes propositional and first-order.

Classical or intuitionistic logic? Both!

Imagine that these two logics fit together in one larger logic. Following Gentzen (LK/LJ), Girard (LU), Liang & M (LKU, PIL).

Modal, temporal, spatial?

I leave these out for now. There is likely to always be a frontier that does not (immediately) fit.

Which proof system?

There are numerous, well studied proof systems: *natural deduction*, *sequent*, *tableaux*, *resolution*, *Herbrand disjunctions*, etc.

Many others are clearly proof-like: *tables* (in model checking), *winning strategies* (in game playing), etc.

Other: *certificates for primality*, etc.

We wish to capture all such proof evidence.

Of course, handling so many proof formats might make for a terribly complex proof checker.

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 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 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) are built on LCF.

Atoms and molecules of inference

We outline how all these demands on certificates can be addressed using what we know of the theory of proof structures.

There are **atoms of inference**.

- Gentzen's **sequent calculus** first provided these: introduction and structural rules.
- Girard's **linear logic** refined our understanding of these further.
- To account for first-order structure, we also need **fixed points** and **equality**.

There are **molecules of inference**.

- There are "rules of chemistry" for assembling atoms of inference into molecules of inference ("synthetic inference rules").

Satisfying the desiderata

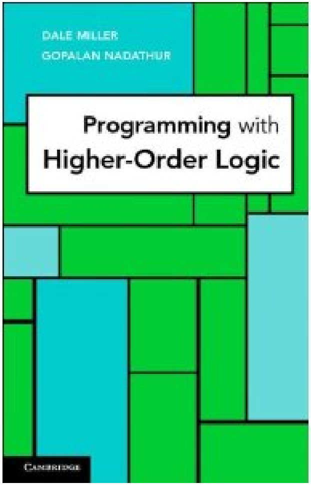
D1: Simple checkers.
Only the atoms of inference and the rules of chemistry (both small and closed sets) need to be implemented in the checker.

D2: Certificates supports a wide range of proof systems.
The molecules of inference can be engineered into a wide range of existing inference rules.

D3: Certificates are based on proof theory.
Immediate by design.

D4: Details can be elided.
Search using atoms will match search in the space of molecules, ie., don't invent new molecules in the checker.

Safe proof reconstruction via logic programming



Logic programming can check proofs in sequent calculus.

Proof reconstruction requires unification and (bounded) proof search.

The λProlog programming language [M & Nadathur, 1986, 2012] also include types, abstract datatypes, and higher-order programming.

Outline

Invertibility of inference rules

Consider a one-side sequent calculus system for classical logic.

Some introduction rules are *invertible*.

$$\frac{\vdash \Delta, B[y/x]}{\vdash \Delta, \forall x B} \qquad \frac{\vdash \Delta, B_1, B_2}{\vdash \Delta, B_1 \vee B_2} \qquad \frac{\vdash \Delta, B_1 \quad \vdash \Delta, B_2}{\vdash \Delta, B_1 \wedge B_2}$$

Some introduction rules are *not invertible*.

$$\frac{\vdash \Delta, B[t/x]}{\vdash \Delta, \exists x B} \qquad \frac{\vdash \Delta, B_i}{\vdash \Delta, B_1 \vee B_2}\ i \in \{1, 2\}$$

Theorem proving researcher often use the invertible inference rules. But invertibility comes with a high cost in copying.

A certificates for propositional logic: compute CNF

Consider only invertible introduction rules for these connectives.

A proof of formula B has a single, huge invertible part.

$$\dfrac{\dfrac{\dots \qquad \dfrac{}{\vdash L_1, \dots, L_n}\ \mathit{closed} \qquad \dots}{\vdots}}{\vdash\ B}$$

A premise is closed if $L_i = \neg L_j$ for $\{i, j\} \subseteq \{1, \dots, n\}$.

The proof certificate can specify these complementary literals for each premise or it can ask the checker to *search* for them.

Proof certificates can be tiny but require exponential time for checking.

Non-invertible rules allow for inserting information

Let B be a large propositional formula and let

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

for propositional variable p. Using non-invertible rules allows for "cleverness" to be injected into the proof.

$$\begin{array}{cl}
\hline
\vdash (p \vee B) \vee \neg p, \neg p,\ p & \textit{closed} \\
\hline\hline
\vdash (p \vee B) \vee \neg p, \neg p,\ (p \vee B) \vee \neg p & * \\
\hline
\vdash (p \vee B) \vee \neg p, \neg p & \\
\hline\hline
\vdash (p \vee B) \vee \neg p, (p \vee B) \vee \neg p & * \\
\hline
\vdash (p \vee B) \vee \neg p &
\end{array}$$

Clever choices $*$ are injected twice. The subformula B is avoided.

LKF : a focused proof systems for classical logic

$$\frac{}{\vdash \Theta \Uparrow \Gamma, t^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, A \qquad \vdash \Theta \Uparrow \Gamma, B}{\vdash \Theta \Uparrow \Gamma, A \wedge^- B} \qquad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, f^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, A, B}{\vdash \Theta \Uparrow \Gamma, A \vee^- B}$$

$$\frac{}{\vdash \Theta \Downarrow t^+} \qquad \frac{\vdash \Theta \Downarrow \Gamma_1, B_1 \qquad \vdash \Theta \Downarrow \Gamma_2, B_2}{\vdash \Theta \Downarrow \Gamma_1, \Gamma_2, B_1 \wedge^+ B_2} \qquad \frac{\vdash \Theta \Downarrow \Gamma, B_i}{\vdash \Theta \Downarrow \Gamma, B_1 \vee^+ B_2}$$

Init Store Release Decide

$$\frac{}{\vdash \neg P_a, \Theta \Downarrow P_a} \qquad \frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, C} \qquad \frac{\vdash \Theta \Uparrow \mathcal{N}}{\vdash \Theta \Downarrow \mathcal{N}} \qquad \frac{\vdash \mathcal{P}, \Theta \Downarrow \mathcal{P}}{\vdash \mathcal{P}, \Theta \Uparrow \cdot}$$

$\mathcal{P}$ multiset of positives; $\mathcal{N}$ multiset of negatives;
P_a positive literal; 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. B is a tautology if and only if $\hat{B}$ has an LKF proof. [Liang & M, TCS 2009]

Thus 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).

An example

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

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

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

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

Example: Resolution as a proof certificate

A *clause:* $\forall x_1 \ldots \forall x_n [L_1 \vee \cdots \vee L_m]$
A *negated clause:* $\exists x_1 \ldots \exists x_n [L_1 \wedge \cdots \wedge L_m]$

1. A clause C is *trivial* if it contains complementary literals.
2. A clause C_1 *subsumes* C_2 if there is a substitution instance of the literals in C_1 which is a subset of the literals in C_2.
3. 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.

Polarize using $\vee^-$ and $\wedge^+$ (multiplicative connectives).

Let $\vdash_d \Theta \Uparrow \Gamma$ mean that $\vdash \Theta \Uparrow \Gamma$ has a proof with decide depth d.

- If C is trivial then $\vdash_1 \cdot \Uparrow C$.
- If C_1 subsumes a non-trivial clause C_2 then $\vdash_2 \neg C_1 \Uparrow C_2$.
- If C_3 is a resolvent of C_1 and C_2 then $\vdash_3 \neg C_1, \neg C_2 \Uparrow C_3$.

Example: Resolution as a proof certificate (cont)

Translate a refutation of $C_1, \ldots, C_n$ into an LKF proof with small holes as follows: assume that $\{i, j\} \subseteq \{1, \ldots, n\}$ and that a resolvent of C_i and C_j is C_{n+1}.

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

Here, Ξ can be replaced with a "hole" annotated with decide depth bound 3.

Future directions

Define many more proof certificate for first-order logic.

- We need to provide for their modular construction.

Improve performance of checking.

Develop focused proof systems for fixed points (recursive definitions).

- This will allow model checkers and inductive theorem provers to share proofs.
- What is a good proof search mechanism to check these? λProlog will not work.

Generalize "proof certificates" to include both *partial proofs* and *counter-examples*. Both have economic and didactic value.

Get certificates adopted. Start with prover competitions?

Some recent references

[1] Chihani, M, and Renaud. Foundational proof certificates in first-order logic. Submitted.

[2] Liang and M. Focusing and polarization in linear, intuitionistic, and classical logics. *TCS*, 2009.

[3] Liang and M. *Kripke Semantics and Proof Systems for Combining Intuitionistic Logic and Classical Logic*, APAL 2013.

[4] M, A proposal for broad spectrum proof certificates, *CPP 2011: Certified Proofs and Programs*.

[5] Nigam and M. A framework for proof systems, *J. of Automated Reasoning*, 2010.

First-order terms and their structure

$$\frac{\vdash \Theta \Uparrow \Gamma, A[y/x]}{\vdash \Theta \Uparrow \Gamma, \forall x\ A}\ \S \qquad \frac{\vdash \Theta \Downarrow \Gamma, A[t/x]}{\vdash \Theta \Downarrow \Gamma, \exists x\ A}$$

$\S$ y is not free in the lower sequent

$$\frac{}{\vdash \Theta \Downarrow t = t} \qquad \frac{}{\vdash \Theta \Uparrow \Gamma, s \neq t}\ \ddagger \qquad \frac{\vdash \Theta\sigma \Uparrow \Gamma\sigma}{\vdash \Theta \Uparrow \Gamma, s \neq t}\ \dagger$$

$\ddagger$ s and t are not unifiable. $\dagger$ s and t have mgu σ.

$$\frac{\vdash \Theta \Uparrow \Gamma, B(\nu B)\bar{t}}{\vdash \Theta \Uparrow \Gamma, \nu B\bar{t}} \qquad \frac{\vdash \Theta \Downarrow \Gamma, B(\mu B)\bar{t}}{\vdash \Theta \Downarrow \Gamma, \mu B\bar{t}}$$

B is a formula with $n \geq 0$ variables abstracted; $\bar{t}$ is a list of n terms.

Here, μ and ν denotes some fixed point. Least and greatest require induction and co-induction.

Examples of fixed points

Natural numbers: terms over 0 for zero and s for successor. Two ways to define predicates over numbers.

$$\begin{array}{rcl} nat\ 0 & \text{:-} & true. \\ nat\ (s\ X) & \text{:-} & nat\ X. \\ leq\ 0\ Y & \text{:-} & true. \\ leq\ (s\ X)\ (s\ Y) & \text{:-} & leq\ X\ Y. \end{array}$$

Above, as a logic program and below, as fixed points.

$$nat = \mu(\lambda p \lambda x.(x = 0) \vee^+ \exists y.(s\ y) = x \wedge^+ p\ y)$$

$$leq = \mu(\lambda q \lambda x \lambda y.(x = 0) \vee^+ \exists u \exists v.(s\ u) = x \wedge^+ (s\ v) = y \wedge^+ q\ u\ v).$$

Horn clauses can be made into fixed point specifications.

The engineering of proof systems

Consider proving the down-arrow focused sequent

$$\vdash \Theta \Downarrow (leq\ m\ n \wedge^+ N_1) \vee^+ (leq\ n\ m \wedge^+ N_2),$$

where m, n are natural numbers and N_1, N_2 are negative formulas. There are exactly two possible macro rules:

$$\frac{\vdash \Theta \Downarrow N_1}{\vdash \Theta \Downarrow (leq\ m\ n \wedge^+ N_1) \vee^+ (leq\ n\ m \wedge^+ N_2)} \text{ for } m \leq n$$

$$\frac{\vdash \Theta \Downarrow N_2}{\vdash \Theta \Downarrow (leq\ m\ n \wedge^+ N_1) \vee^+ (leq\ n\ m \wedge^+ N_2)} \text{ for } n \leq m$$

A macro inference rule can contain an entire Prolog-style computation.

The engineering of proof systems (cont)

Consider proofs involving simulation.

$$sim\ P\ Q \equiv \forall P' \forall A[\ P \xrightarrow{A} P' \supset \exists Q'\ [Q \xrightarrow{A} Q' \wedge sim\ P'\ Q']].$$

Typically, $P \xrightarrow{A} P'$ is given as a table or as a recursion on syntax (*e.g.*, CCS): hence, as a fixed point.

The body of this expression is exactly two “macro connectives”.

- $\forall P' \forall A[P \xrightarrow{A} P' \supset \cdot\ ]$ is a negative “macro connective”. There are no choices in expanding this macro rule.
- $\exists Q'[Q \xrightarrow{A} Q' \wedge^+ \cdot\ ]$ is a positive “macro connective”. There can be choices for continuation Q'.

These macro-rules now match exactly the sense of simulation from model theory / concurrency theory.