&Truth

Standing on the Shoulders of Giants

A Letter Acknowledging the Legacy that we are Building on

by Clark Barrett

October 29, 2026

For hundreds of years, philosophers and mathematicians have dreamed of being able to delegate difficult reasoning tasks to algorithms and machines. Gottfried Leibniz envisioned a universal symbolic language which could represent every abstract idea with mathematical precision. David Hilbert proposed constructing a general algorithm for determining mathematical truth. And even when results from the likes of Church, Turing, and Gödel showed that full universality could not be achieved, the ambition to automate as much as possible remained.

The rise of computers and computing, then, was naturally accompanied by a desire to reason about computation. Luminaries like Robert Floyd, Tony Hoare, Edsger Dijkstra, and Amir Pnueli pioneered logics for this purpose, laying the foundation for program verifiers. Another crucial piece came from the programming languages community, where key figures like Dana Scott, Christopher Strachey, and Gordon Plotkin introduced notions of denotational and operational semantics, providing a way to interpret statements in a program as mathematical objects. Finally, pioneers like Roger Hindley and Robin Milner, Jean-Yves Girard and John Reynolds, and Thierry Coquand and Gérard Huet devised the foundational type systems that would later be needed to build general-purpose frameworks for modeling and automating code reasoning.

Inspired and enabled by these foundations, builders developed tools and techniques for assisting humans with reasoning tasks. Chief among such efforts was the creation of interactive theorem provers, systems designed to manage and check large mathematical reasoning efforts. These leveraged the observation by Haskell Curry and William Howard that there is a correspondence between programs and proofs and between types and theorems. Robin Milner's LCF contributed two additional important ideas: the trusted kernel with an abstract type of theorems, to ensure that only trusted components could construct theorems; and tactics, small programs for decomposing and solving proof goals. Every modern theorem prover descends from LCF. The modern set of theorem provers includes Robert Boyer and J Moore's ACL2, Sam Owre, Natarajan Shankar and John Rushby's PVS, Mike Gordon and Tom Melham's HOL, John Harrison's HOL Light, Larry Paulson and Tobias Nipkow's Isabelle/HOL, Leo de Moura's Lean, and Coquand and Huet's Rocq, among others. &Truth's technology is built on Lean, and we work closely with the Lean Focused Research Organization to support Lean and its community.

Though powerful, interactive theorem provers require time and extensive expertise to use. An alternative approach is to build narrow, customized tools for reasoning about programs, designed to be more automatic. Early tools included David Luckham's Stanford Pascal Verifier as well as the Extended Static Checker (ESC), initially pioneered by Rustan Leino and Greg Nelson. The back end of ESC used a prover called Simplify, developed by Nelson and Leino together with David Detlefs and James Saxe. Simplify later inspired work on a new generation of provers powered by solvers for satisfiability modulo theories (SMT). SMT solvers pushed the boundaries of automation, identifying fragments of logic that could be solved efficiently in a push-button manner. Modern SMT solvers include Aina Niemetz and Mathias Preiner's Bitwuzla, Alessandro Cimatti, Alberto Griggio, and Roberto Sebastiani's MathSAT, Natasha Sharygina's OpenSMT, Erika Ábrahám's SMT-RAT, Pascal Fontaine's veriT, de Moura and Bruno Dutertre's Yices, and de Moura and Nikolaj Bjørner's Z3, as well as cvc5, developed by my group together with Cesare Tinelli, Andrew Reynolds, Haniel Barbosa, and Yoni Zohar. This seems a suitable moment to also acknowledge my own PhD advisor, David Dill, who in addition to making many of his own contributions to formal methods, played a crucial role in both the genesis of SMT solving and in shaping me as a thinker and a scientist. I consider it a rare privilege that I have continued to be able to work closely with him in recent years and to benefit from his mentorship.

Modern program verifiers backed by SMT solvers include CN (Christopher Pulte and Peter Sewell among others) and F* (Nikhil Swamy and team) for C code; Leino's Dafny (both the name of the verifier and the language); SPARK for Ada (Rod Chapman and others); Stainless for Scala (by Viktor Kuncak's group); and the Verus tool for Rust (by Andrea Lattuada, Travis Hance, Chris Hawblitzel, Bryan Parno, and Jon Howell among others). Such tools provide inspiration for &Truth's proof automation, which allows us to prove the vast majority of theorems cheaply and efficiently.

One thorny challenge that repeatedly raises its head in all of these verifiers is that of reasoning about the heap. Standard logical encodings of programs as implemented in the solvers mentioned above, often use a flat memory model, in which all of memory is modeled as a single array. While correct in theory, in practice such a model is difficult to use because it requires maintaining a quadratic set of constraints stating that the various objects represented in memory are in disjoint memory locations. An early attempt to mitigate this was Rod Burstall's type-based model of memory, but the problems remained. An ingenious solution to this problem emerged with the introduction of separation logic, developed by John Reynolds, Peter O'Hearn, Samin Ishtiaq and Hongseok Yang. The separating conjunction of separation logic ensures disjointness by construction, meaning the quadratic set of disjointness constraints can be eliminated. Later, O'Hearn and Stephen Brookes made the observation that separation could also be used to effectively reason about concurrency. And Ralf Jung, Robbert Krebbers, Lars Birkedal, and Derek Dreyer led the development of Iris, which brought many previous ideas together into a logic that was versatile enough to enable program verification of messy real-world code. Many tools have been built on top of separation logic or Iris, including: Smallfoot (Josh Berdine, Cristiano Calcagno, Peter O'Hearn); GRASShopper (Ruzica Piskac, Thomas Wies, and Damien Zufferey); and Diaframe (Ike Mulder, Robbert Krebbers, Herman Geuvers). A new version of Iris for Lean is a crucial building block in the logical foundations of &Truth's tools.

The ideas and tools listed above also benefited immensely from thoughtful leaders and funders in both government and industry. The National Science Foundation, with leadership provided over several decades by Nina Amla and others, consistently funded projects in formal methods that have been crucial to the development of the field, as has been DARPA's leadership with programs like Kathleen Fisher's HACMS, Ray Richards' CASE, and Brad Martin's PROVERS. Industry leaders with

the foresight to invest in and grow the research area include Tom Ball at Microsoft Research; Byron Cook at AWS; John Launchbury, co-founder of Galois; and Darrin Cofer and David Hardin from Collins Aerospace.

Many more individuals than can be named in this brief post, many of them friends and colleagues, have contributed to the body of formal methods knowledge. Their hard work and genius brought the field of formal methods to the point where it is today, where in theory, we have methodologies available that can verify pretty much any reasonable program in any language. &Truth is now helping transition formal methods from possible in theory to possible in practice.

The last obstacle to making formal methods widely applicable was the significant human effort required to use formal tools in a scalable way. The advent in just the last year of AI models with superhuman capabilities to write code and do mathematics provides the solution for overcoming this obstacle. Formal tools can be driven by AI agents instead of humans in an extremely safe and synergistic way. AI proposes specifications and proofs, and formal tools check whether these are valid. The entire cycle can iterate automatically to scalably prove code without the need for human intervention. Recent work done in my lab in collaboration with Livia Sun (my former student, now at &Truth), David Dill, and others, confirms that the inflection point has been reached: AI has the ability to massively improve the automatability and scalability of formal methods.

The &Truth team is honored to acknowledge our debt to the thinkers and visionaries who have provided the deep technical foundations on which our mission rests. We view ourselves as the fortunate inheritors of their intellectual legacy and sharers of their visions and dreams. We hope that the world of safe, secure, and correct code that we aim to help create is one they would be proud of.

Clark Barrett

Co-Founder and Co-CEO, &Truth

Palo Alto, Singapore, October 29, 2026