Mechanical Reasoning Tools and the Augmentation of Human Thought

J Strother Moore

Department of Computer Sciences

University of Texas at Austin

presented at

American Association for the Advancement of Science

Annual Meeting, 2006

St. Louis

16-20 February, 2006

Abstract

Computers are good at “number crunching”, e.g., at simulating the behaviors of complex systems described by sets of differential equations and concretely specified initial conditions. But can they help scientists with abstract reasoning? Can computers reason?

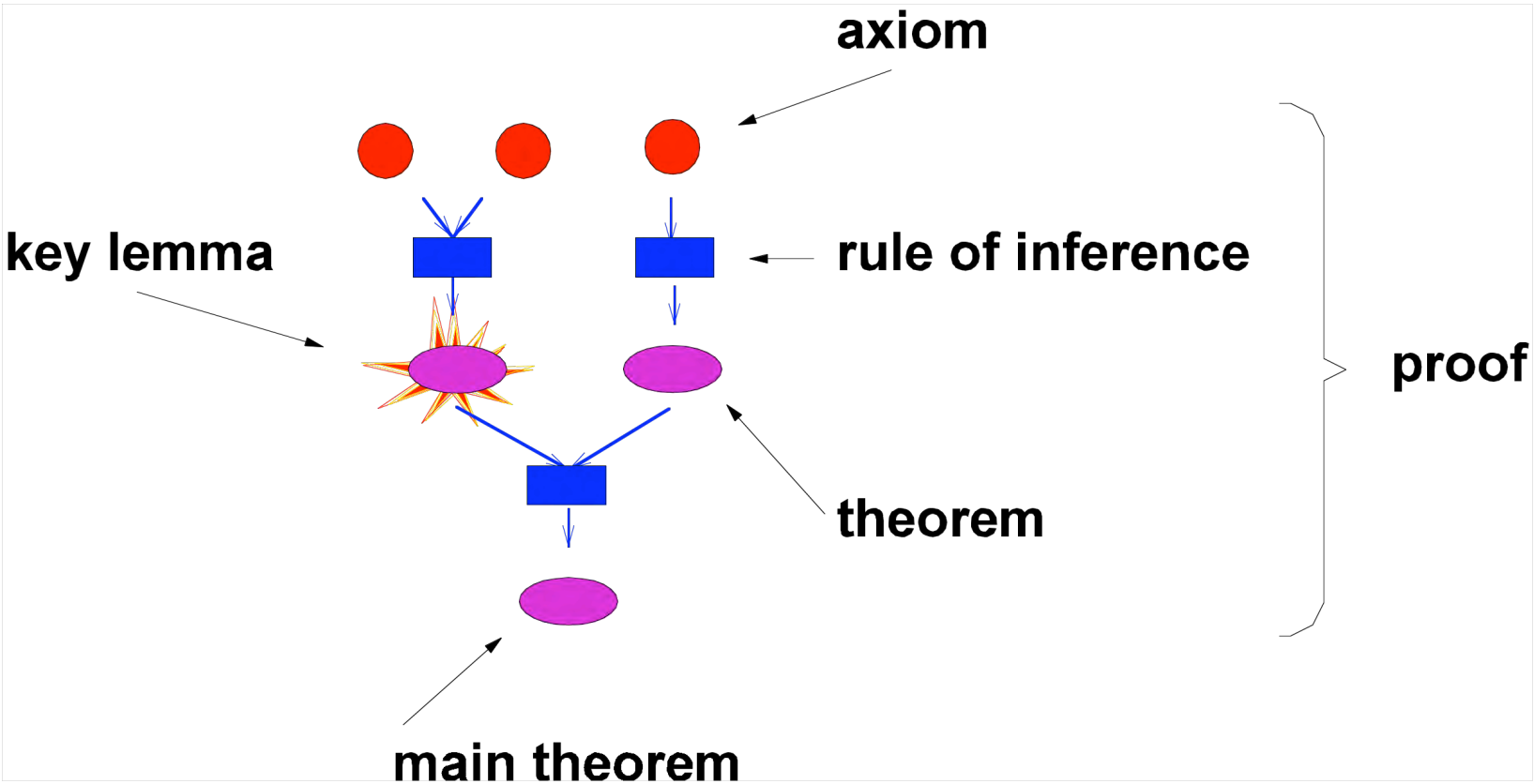
If by “reason” we mean deduce abstract properties of systems described symbolically, the answer is “yes”.

If we had some exact language ... or at least a kind of truly philosophic writing, in which the ideas were reduced to a kind of alphabet of human thought, then all that follows rationally from what is given could be found by a kind of calculus, just as arithmetical or geometrical problems are solved.

— Leibniz (1646 - 1716)

Basic Notions

- **syntax** — specifying formulas
- **axioms** — formulas taken as given
- **rules of inference** — formula transformation rules
- **theorems** — formulas derived from axioms with rules of inference
- **proofs** — the derivations themselves
- **semantics** — determines which formulas are valid ("always true")
- **decidability** — whether there is an algorithm for distinguishing theorems from non-theorems.


axiom
key lemma
rule of inference
proof
theorem
main theorem

If

- the axioms are valid and
- the rules of inference are validity preserving,

then

theorems are valid.

Thus, if you want to know that something is always true, try writing it as a symbolic formula and proving it.

Example Formal Systems

- propositional calculus
- propositional temporal logic
- predicate calculus
- higher-order logic
- set theory
- lambda calculus

Some systems are decidable, others not.

But because proofs are finite syntactic objects, it is possible to create *mechanized reasoning engines* that explore the space of formulas derivable from the axioms.

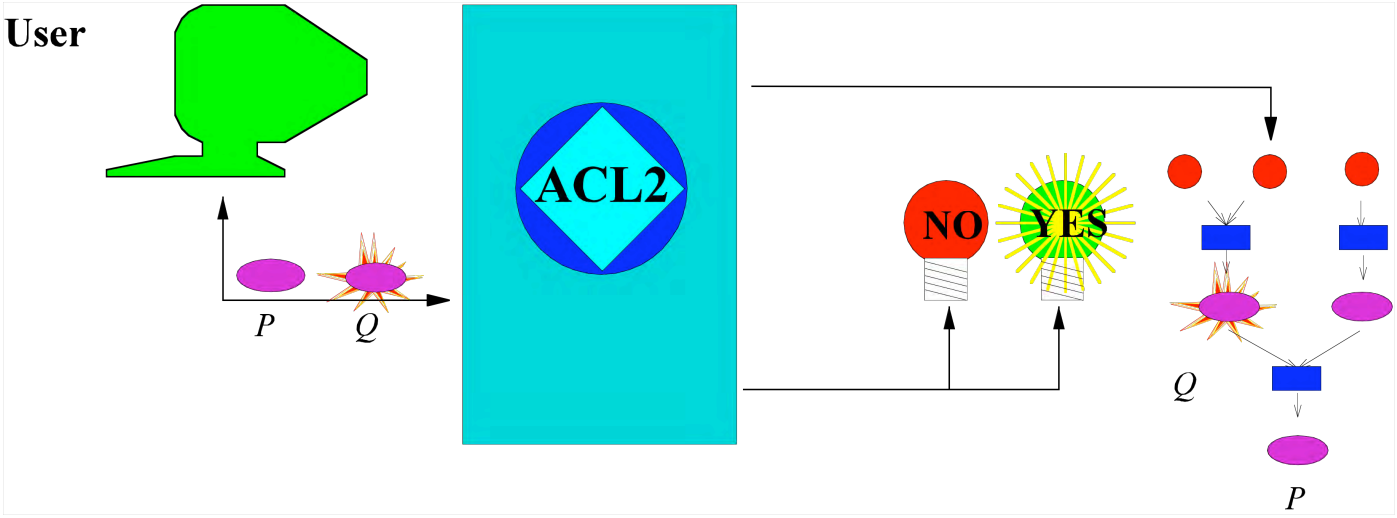
You may think of these engines as similar to chess playing programs, exploring only the "plausible" legal moves.

A reasoning engine **may be**

• *incomplete* — incapable of finding proofs for all theorems,

but **must be**

• *sound* — correct when it says something is a theorem.



By mechanized reasoning engine I include equivalence checkers, satisfiability solvers, model checkers, proof checkers, and interactive theorem provers.

There are a large number of mechanized reasoning engines, including Conformal, zChaff, SMV, Mizar, Nqthm, HOL, ACL2, Maude, PVS, Coq.

Using a Reasoning Engine

- formalize a model of the problem (typically by writing axioms and defining relevant functions and relations)
- formalize a conjecture
- submit the conjecture to the engine
- provide guidance

The first step is often the most important: formalizing the problem at an appropriate level of abstraction can often make it tractable.

Applications in Computer Science

- **equivalence checking** — checking that the gate level design is propositionally equivalent to a higher-level description (tautology checking)

- **model checking** — checking that a given finite state machine satisfies certain propositional properties along all its (relevant) paths

- **theorem proving** — verifying that hardware and software have certain functional properties

Some Numbers

Tautology checking can be done in exponential time: A formula involving n propositional variables can be exhaustively checked in 2^n cases.

Modern symbolic tautology checkers routinely check formulas involving tens of thousands of variables.

But Is Mechanized Aid Really Necessary?

"An elusive circuitry error is causing a chip used in millions of computers to generate inaccurate results" -- *NY Times, "Circuit Flaw Causes Pentium Chip to Miscalculate, Intel Admits", Nov 11, 1994*

"Intel Corp. last week took a $475 million write-off to cover costs associated with the divide bug in the Pentium microprocessor's floating-point unit" --- *EE Times, Jan 23, 1995*

To exhaustively test a 64-bit floating-point divider would take about 2×10^{17} years on a petaflop machine (10^{15} operations per second).

By July, 1995, the microcode for the AMD K5 floating point divide operation had been proved correct using the ACL2 reasoning engine.

A total of 9 weeks were devoted to the project.

All $2^{64} \times 2^{64} \times 18$ cases were handled.

The AMD Athlon FMUL Theorem

```
(defthm correctness-of-fmul
   (let ((ideal (rnd (* (hat x) (hat y))
                     (mode rc)
                     (precision pc)))
         (z (fmul x y rc pc)))
     (implies (and (normal-encoding-p x (extfmt))
                   (normal-encoding-p y (extfmt))
                   (member rc (list 0 1 2 3))
                   (member pc (list 0 1))
                   (repp ideal (extfmt)))
              (and (normal-encoding-p z (extfmt))
                   (= (hat z) ideal)))))
```

An ACL2 proof that a Motorola digital signal processor design implements a given instruction set produced subgoals requiring 5000 pages each ***simply to print.***

The formal proof would consist of many millions of primitive proof steps.

Use of symbolic reasoning engines is qualitatively different from simulation and testing.

Users describe abstract ideas and properties in symbolic terms.

The engines respond in kind, communicating their discoveries or problems in symbolic terms.

Symbolic proofs generally cover an infinity of cases.

Using one of these engines is like having a rather unimaginative colleague at your blackboard who is never too embarrassed to say “I don't understand”, who is never distracted, and who can carry out symbolic processes with blinding speed and total accuracy.

It is now routine at Intel, IBM, AMD, and other microprocessor design companies to analyze certain components (floating point units, instruction decoders, pipelines, memory protocols) with mechanized reasoning engines.

These designs are simply too complex to be analyzed by unaided humans.

Applications from Biochemistry

Researchers at SRI International have used the Maude system to implement **Pathway Logic**.

Pathway Logic is a formal system designed to let biochemists describe certain biological state machines.

Protein “states” are formalized as terms indicating the presence or absence of certain features in the protein.

Biological processes are described by axioms that transform protein states.

The researchers have used this system to model the Epidermal Growth Factor Receptor (EGFR) network, including the Rafl serine-threonine protein kinase.

They are developing two levels of abstraction, the more detailed one differentiating states according to structural features of the protein.

Using Maude's model checker it is possible to ask such questions as ``Can Rafl in a cell described by graf be activated?"

If so, Maude shows a path (sequence of transitions) to the desired state.

If not, the model definitively prohibits Rafl activation.

It is up to the biochemist to determine whether these predictions and thus the models conform to reality.

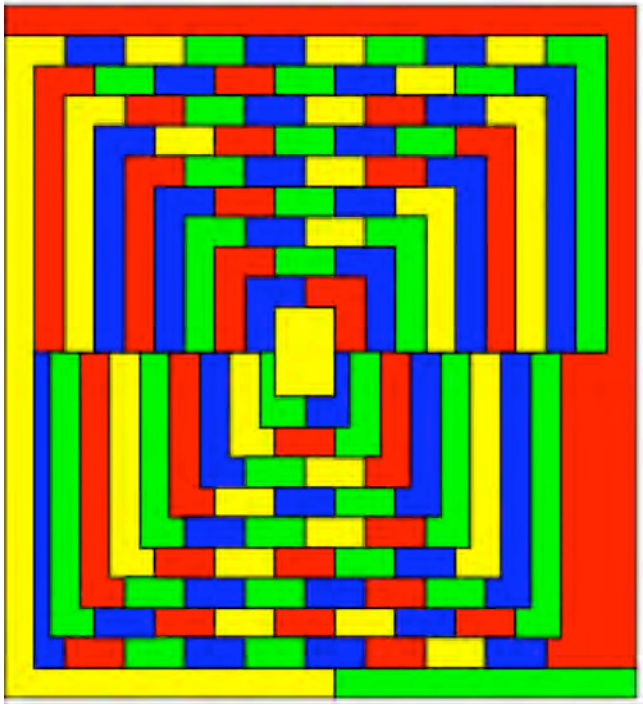
Applications from Mathematics

Gödel's First Incompleteness Theorem: In any consistent formal theory providing basic arithmetic, it is possible to construct a statement that is true but not provable in the theory.

This has often been used to argue that mechanistic reasoning is somehow inferior to human reasoning.

But a machine has proved this theorem with guidance from its human user.

The Four Color Theorem: The regions of any simple planar map can be colored with only four colors so that no two adjacent regions are the same color.



Conjectured 1852.

First “proved” by Appel and Haken in 1976; two controversial aspects:

- manual proof consisting of 10,000combinations of conditions showing the sufficiency of solving a billion special cases

- custom written computer program to dispatch the billion cases.

The first step was later reduced to 2,500 combinations.

In 2004, Gonthier at Microsoft Research Lab, Cambridge (England) and Werner at INRIA (France) used the INRIA Coq mechanized reasoning engine to prove formally that

- the combinatoric analysis was correct, and

- a certain custom-written computer program correctly carried out the analysis of the billion cases.

Finally, after 150 years, the Four Color Theorem has been formally proved and computational reasoning was key.

The Robbins Algebra Problem was open for 60 years. It concerned whether a certain set of axioms characterized all Boolean Algebras.

The proof was found automatically by the EQP theorem prover at Argonne National Labs in 1997.

That was the first instance of a computer program automatically solving a well-studied open problem.

Conclusion

Mechanical reasoning engines are fundamentally expanding what we can reason about.

Today's computers are faster and more reliable in part because scientists and engineers have augmented their abilities to reason about their designs.

Reasoning engines will redefine the colloquial meaning of ***proof***, just as computing has already redefined such words as ***model*** and ***experiment***.

We are approaching a time when fundamental discoveries will be made jointly by humans collaborating and arguing with machines.

Acknowledgements

This talk has touched upon the work of dozens of computer scientists and engineers who have authored and/or used mechanical reasoning systems.

Please see the bibliography of the accompanying paper for appropriate scholarly citations.

Thank you.

J Strother Moore
University of Texas at Austin