Machines Reasoning About Machines

Part 3

How To Drive ACL2

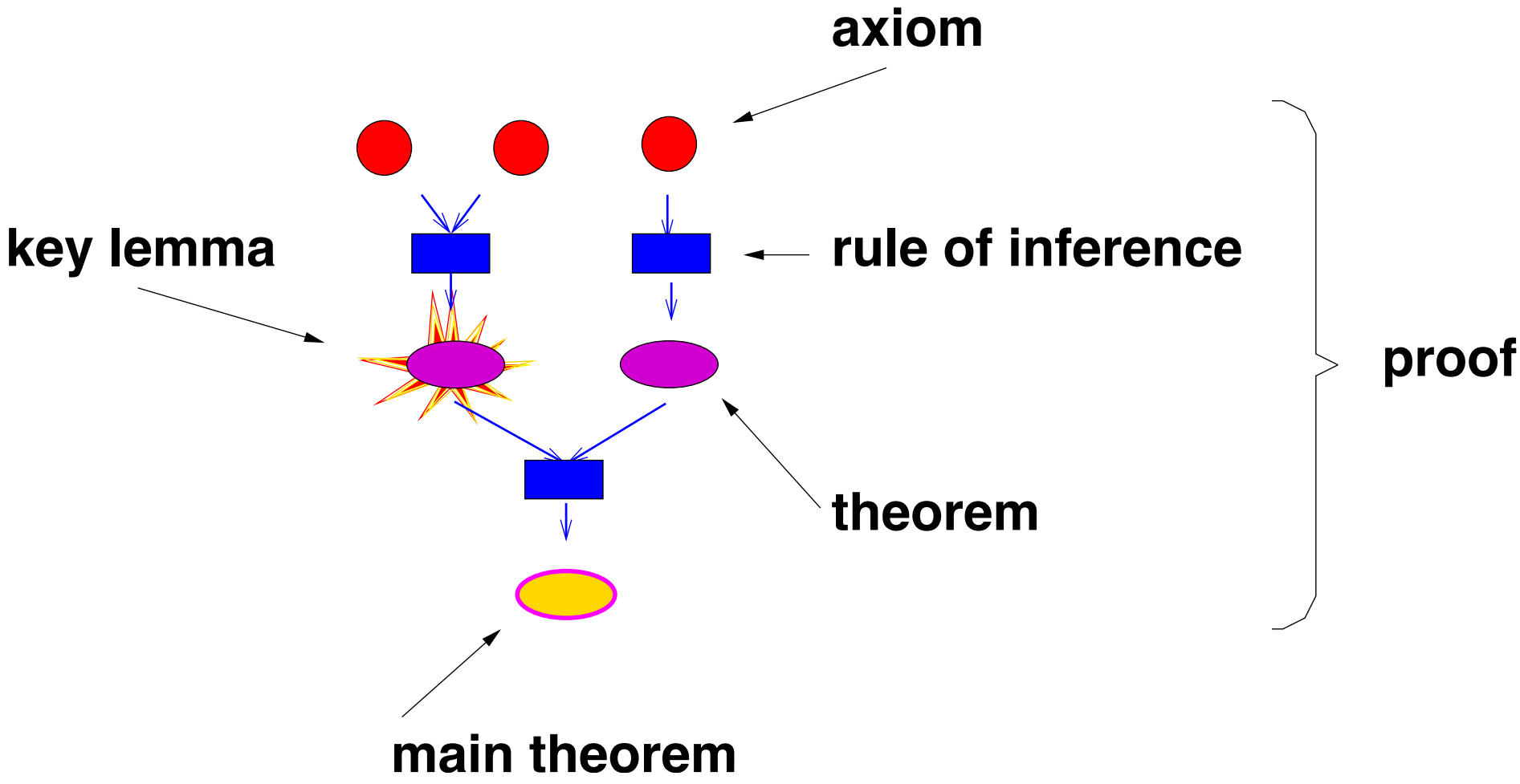
J Strother Moore
Department of Computer Sciences
University of Texas at Austin

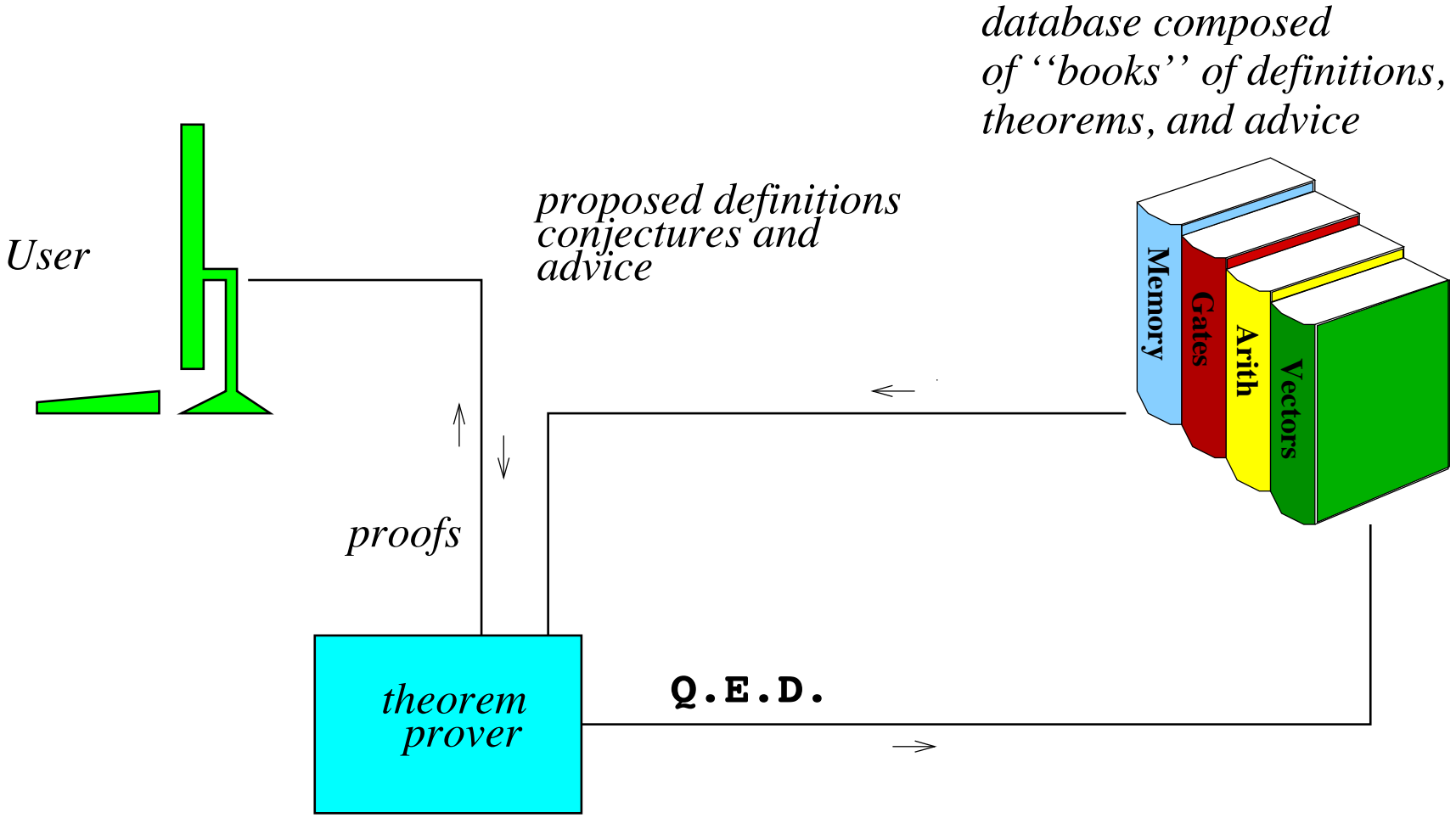
The Setting

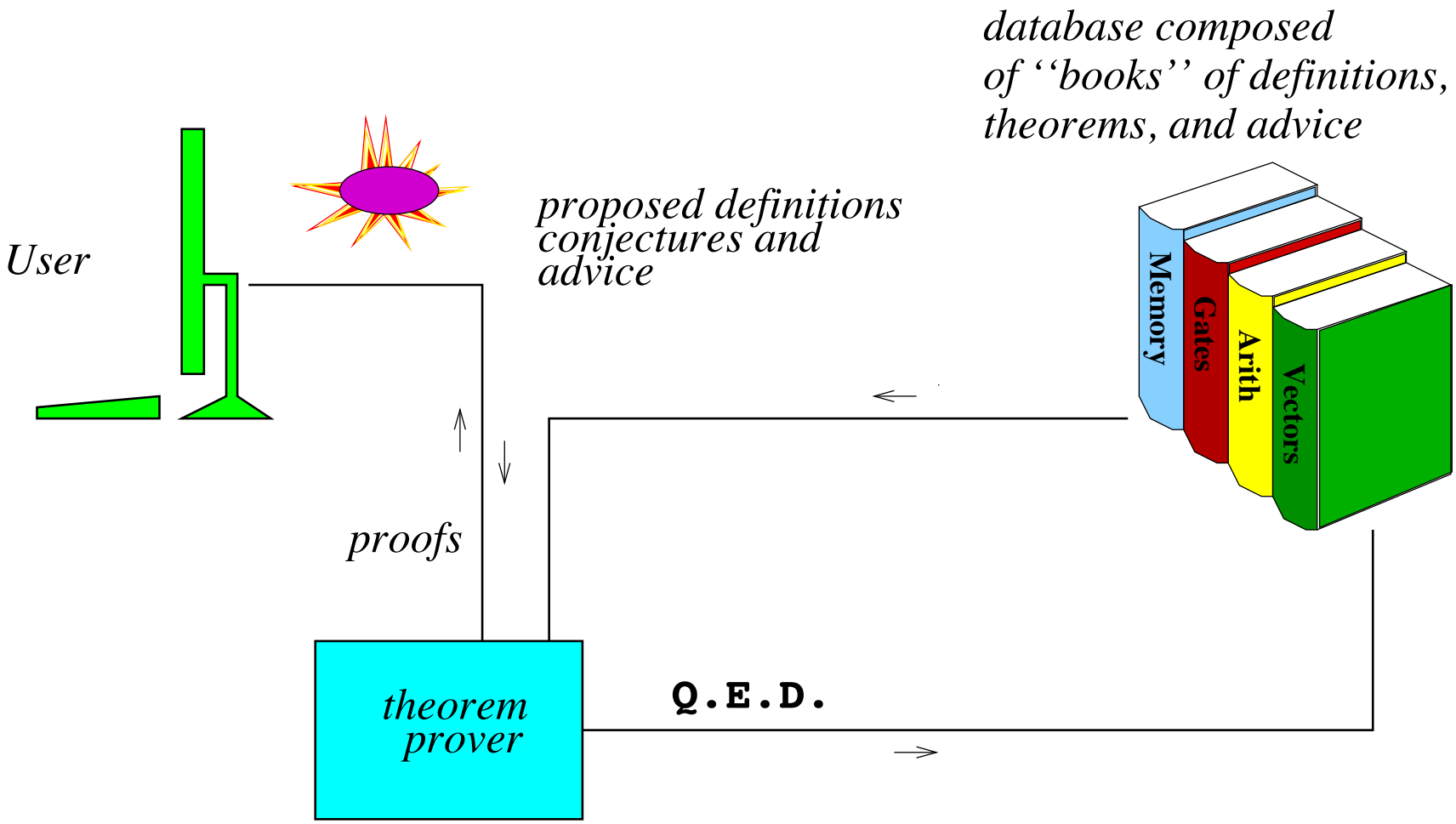
Think of ACL2 as an *assistant* in the proof discovery process.

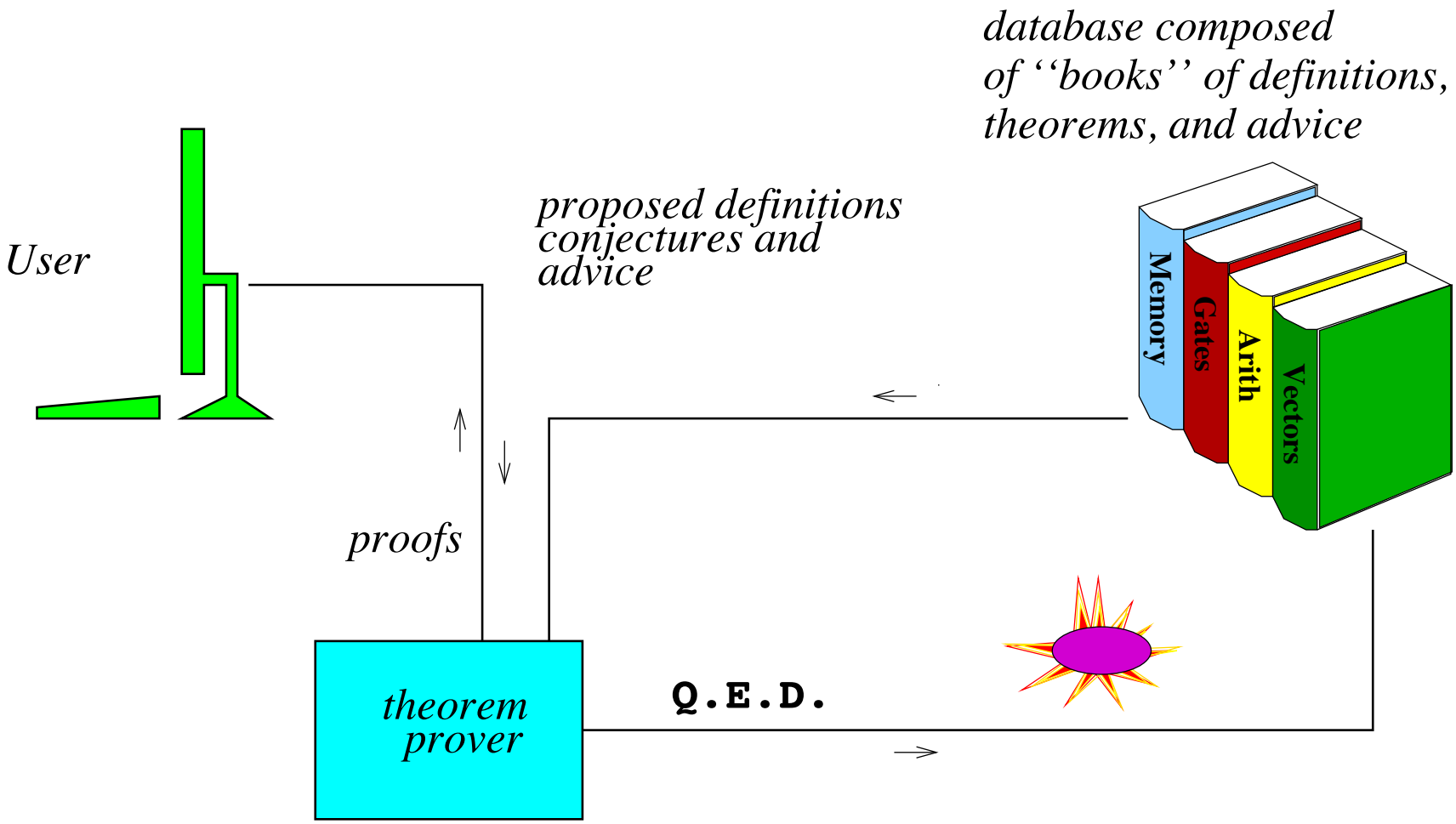
Your role is the creative one.

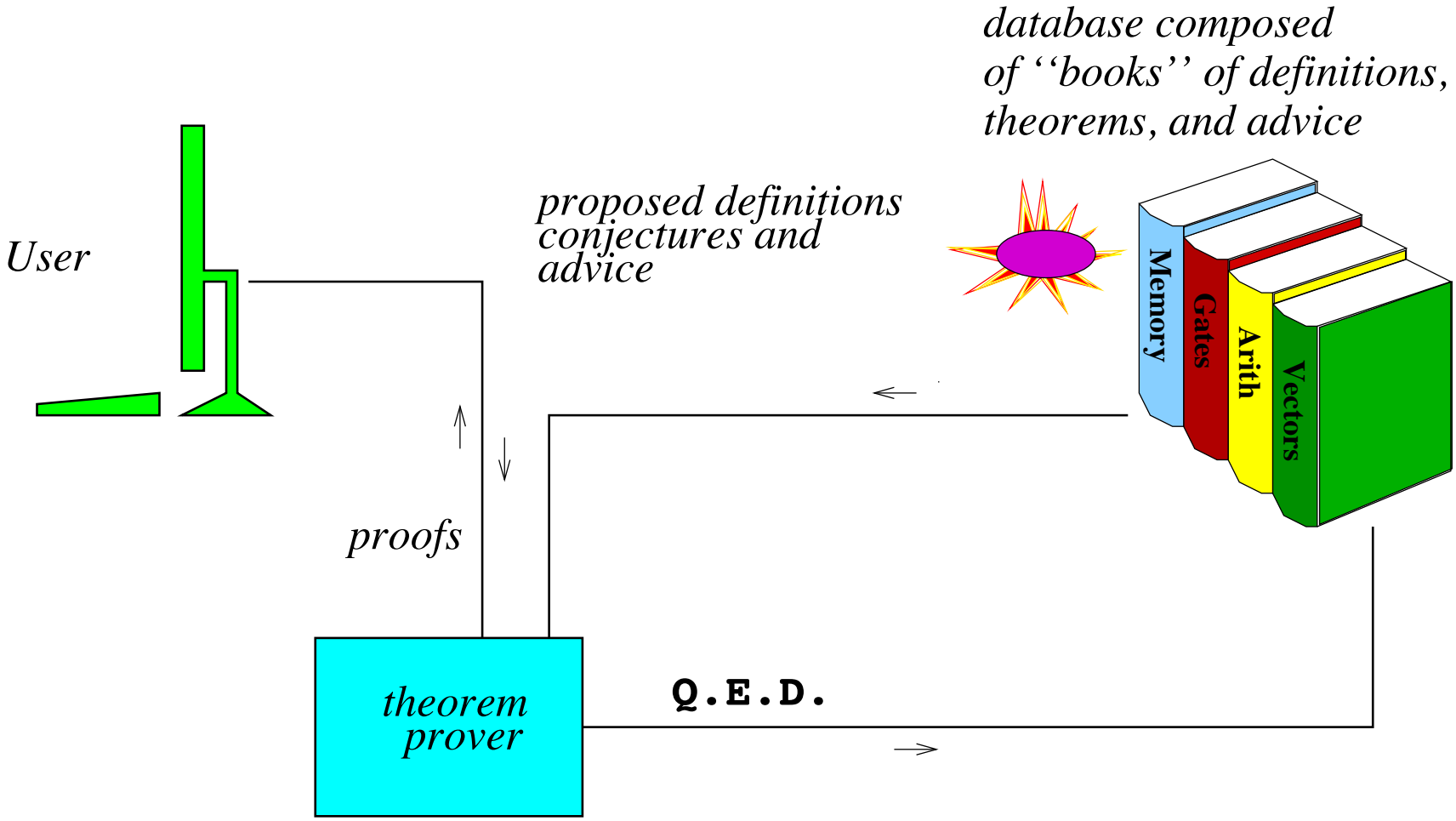
ACL2's role is performing accurate, mechanical transformations using the mathematical truths you reveal to it.


axiom
key lemma
rule of inference
proof
theorem
main theorem


database composed
of ‘‘books’’ of definitions,
theorems, and advice
User
proposed definitions
conjectures and
advice
Memory
Gates
Arith
Vectors
proofs
theorem
prover
Q.E.D.


database composed
of ‘‘books’’ of definitions,
theorems, and advice
User
proposed definitions
conjectures and
advice
Memory
Gates
Arith
Vectors
proofs
theorem
prover
Q.E.D.


database composed
of ‘‘books’’ of definitions,
theorems, and advice
User
proposed definitions
conjectures and
advice
Memory
Gates
Arith
Vectors
proofs
theorem
prover
Q.E.D.


database composed
of ‘‘books’’ of definitions,
theorems, and advice
proposed definitions
conjectures and
advice
User
Memory
Gates
Arith
Vectors
proofs
theorem
prover
Q.E.D.

"The Method" to Prove α

Ask ACL2 to prove α.

When it fails, look at its *Key Checkpoints* and ask:

> *Do I know something, β, that will simplify these formulas?****

If so, use The Method to prove β, then start over to prove α.

***** Major Footnote:**

Sometimes β takes the form of a generalization or correction to α!

Sometimes it takes the form of a decomposition of the proof strategy, e.g., to prove $\alpha_1 \rightarrow \alpha_2$ it might be best to prove $\alpha_1 \rightarrow \gamma$ and $\gamma \rightarrow \alpha_2$.

Plan

I will use The Method to prove a simple theorem.

Then I'll pose another and we'll prove it together.

Problem 1

Let `(rev x)` be the reverse of the list `x`.

Prove that `(rev x)` has duplicate elements precisely if `x` has duplicate elements.

Demo 1

Solution 1

```
Lemma 2:   (mem e (ap a b)) = (or (mem e a)
                                  (mem e b))

Lemma 1:   (has-dups (ap a (list e)))
           =
           (or (mem e a)
               (has-dups a))

Lemma 3:   (mem e (rev x)) = (mem e x)

Problem 1: (has-dups (rev x)) = (has-dups x)
```

Problem 2 In this problem we'll study two different ways to implement the idea of collecting one copy of every element in a list.

But since order matters, the two different algorithms produce different (but "equivalent") answers.

Problem 2

Define the function `collect-last` to collect the last occurrence of each element.

Define `collect-first` to collect an element if it has not already been collected.

State and prove the relationship between them.

Demo 2

Questions?

Thank You

I really appreciate that you gave me 3 hours of your time.

Thank you very much.