Network Security Analysis via Predicate Logic

&

Increasing Trustworthiness: A case study in Theorem Prover Design

CS3234

Lecture 7

Aquinas Hobor and Martin Henz

Network Security Analysis via Predicate Logic

Process for applying theory to practice

1. Learn about problem

2. Create a formal model of the problem

3. State the goal

4. Use some kind of tool (theorem prover, SAT solver, etc.) to solve

Process for applying theory to practice

1. Learn about problem
2. Create a formal model of the problem
3. State the goal
4. Use some kind of tool (theorem prover, SAT solver, etc.) to solve

Problem

- We have a network of many computers (100s-1,000s-10,000s)
- Each computer only allows certain kinds of connections (example: the accounting computer only allows the CEO's computer to access it; anyone in the world can access the http services of the web server)
- Each computer is running different kinds of software
  - Mail software
  - Sales software
  - Office software
  - Web hosting software
  - etc.
- Often different computers are running different versions, different patches, etc.

Problem

We wish to guarantee some security policy, such as:

- Only the CEO can access at the accounting data

How can we try to do this?

Fact: most security breaches are exploits of known vulnerabilities. Defending against truly new vulnerabilities is really hard, so let's concentrate on the common case.

Process for applying theory to practice

1. Learn about problem

2. Create a formal model of the problem

3. State the goal

4. Use some kind of tool (theorem prover, SAT solver, etc.) to solve

Why do you take CS courses?

In this class, we are teaching you a set of tools

- Propositional Logic
- SAT Solving
- Natural Deduction
- Theorem Proving
- Predicate Logic
- Modal Logic
- Temporal Logic
- Model Checkers
- Hoare Logic

Why do you take CS courses?

In this class, we are teaching you a set of tools

- Propositional Logic ✓
- SAT Solving ✓
- Natural Deduction ✓
- Theorem Proving ✓
- Predicate Logic ✓
- Modal Logic
- Temporal Logic
- Model Checkers
- Hoare Logic

Learning the tools is not easy…

Learning the tools is not easy…

… as you know from the homework and exam…

Learning the tools is not easy…

… as you know from the homework and exam…

… but figuring out which tools can help in which situations is <u>hard</u> (knowing the tools well is a prerequisite, which is why you take courses…)

Usually you have to study a problem for some time before you get a good idea.

Model

- We will model the network with a series of implications (essentially how an attacker would break our policy)

- We have two basic classes of rules:
  - Network topology
  - Attack vulnerability

- Example rules (network topology):
  - forall (p : computer), AccessHTTP(p, WebServerComputer)
  - …
  - RunningApache1.0(WebServerComputer)
  - …

More rules

- Attack vulnerability rule:
  - …
  - KnownAttack42: forall (p1 : computer) (p2 : computer), RunningApache1.0(p2) -> AccessHTTP(p1,p2) -> TakeOver(p1,p2)
  - …

Uh oh…

It appears that anyone can take over the webserver!

More rules

– …

– TakeOver(CEOComputer, AccountingComputer)

– …

The CEO likes direct access to the accounting computer so that he can see the latest sales results.

More rules

- …
- AccessReportTool(WebServerComputer, CEOComputer)
- …

The CEO likes to get regular reports and statistics from his webserver, so he uses AccessReportTool, which is this really great piece of software, to do this.

More rules

- …
- KnownAttack212: forall p1 p2,
  AccessReportTool(p1,p2) -> TakeOver(p1,p2)
- …

Unfortunately, he downloaded it from a hacker website...

How to hack the accounting computer (and why an evildoer would want to)

1. Access the webserver:
   - forall (p : computer), AccessHTTP(p, WebServerComputer)

2. Since the webserver is running an old version of Apache, take it over:
   - RunningApache1.0(WebServerComputer)
   - KnownAttack42: forall (p1 : computer) (p2 : computer), RunningApache1.0(p2) -> AccessHTTP(p1,p2) -> TakeOver(p1,p2)

3. Since the CEO is nice enough to have installed AccessReportTool and let it access his machine, use it to take it over:
   - AccessReportTool(WebServerComputer, CEOComputer)
   - KnownAttack212: forall p1 p2, AccessReportTool(p1,p2) -> TakeOver(p1,p2)

4. Since the CEO likes direct access to the accounting computer, you can now take over the accounting computer
   - TakeOver(CEOComputer, AccountingComputer)

5. Transfer money to secret bank account

6. Flee country

Process for applying theory to practice

1. Learn about problem

2. Create a formal model of the problem

3. State the goal

4. Use some kind of tool (theorem prover, SAT solver, etc.) to solve

Goal

What you want to show is that:

forall p, p <> CEOComputer ->

~TakeOver(p, AccountingComputer)

This is one way to formally state the policy; as the policy gets more complicated it gets harder to state it…

Process for applying theory to practice

1. Learn about problem

2. Create a formal model of the problem

3. State the goal

4. Use some kind of tool (theorem prover, SAT solver, etc.) to solve

5. Building a business…

- Network Topology
  - Which connections different computers accept
  - This must be determined by some kind of network analysis tool, maybe that you run each night

- Known Attacks
  - Distributed by some security firm (think antivirus software)

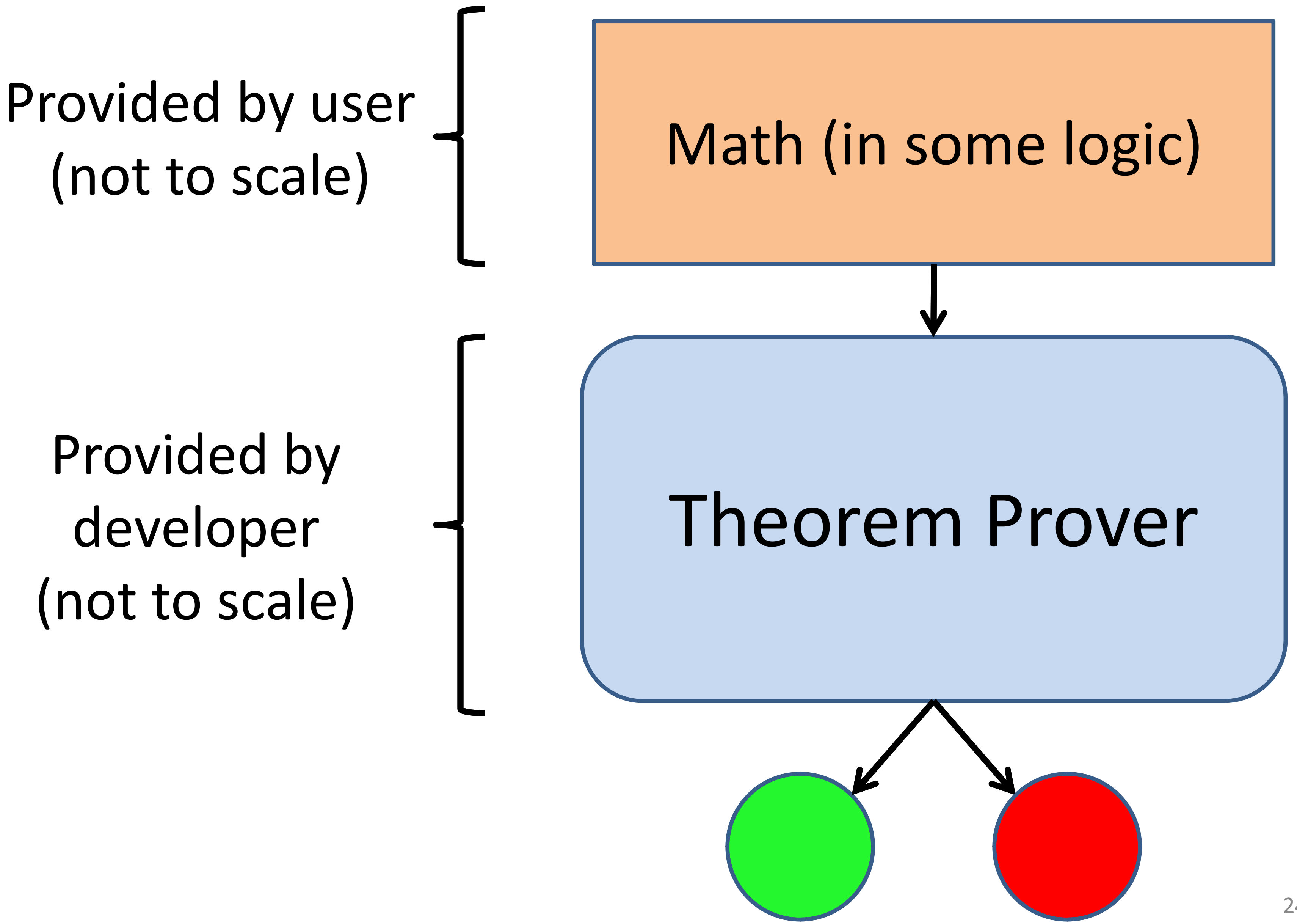
(unfortunately, other people have already patented this idea…)

Something completely different…

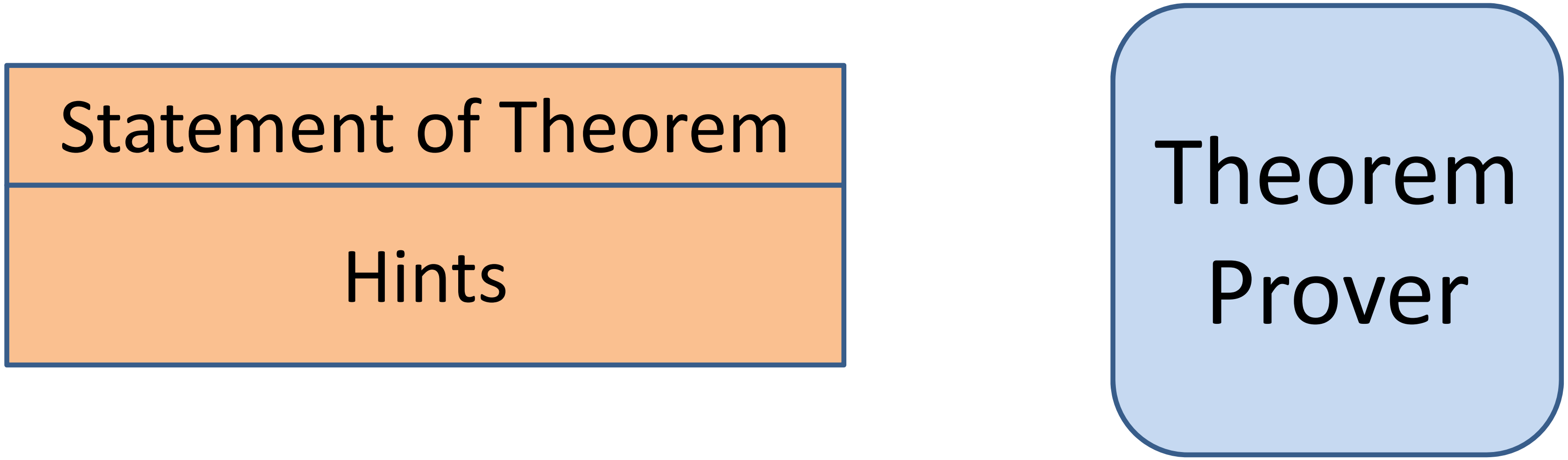
How do we build a trustworthy system?

(a case study)

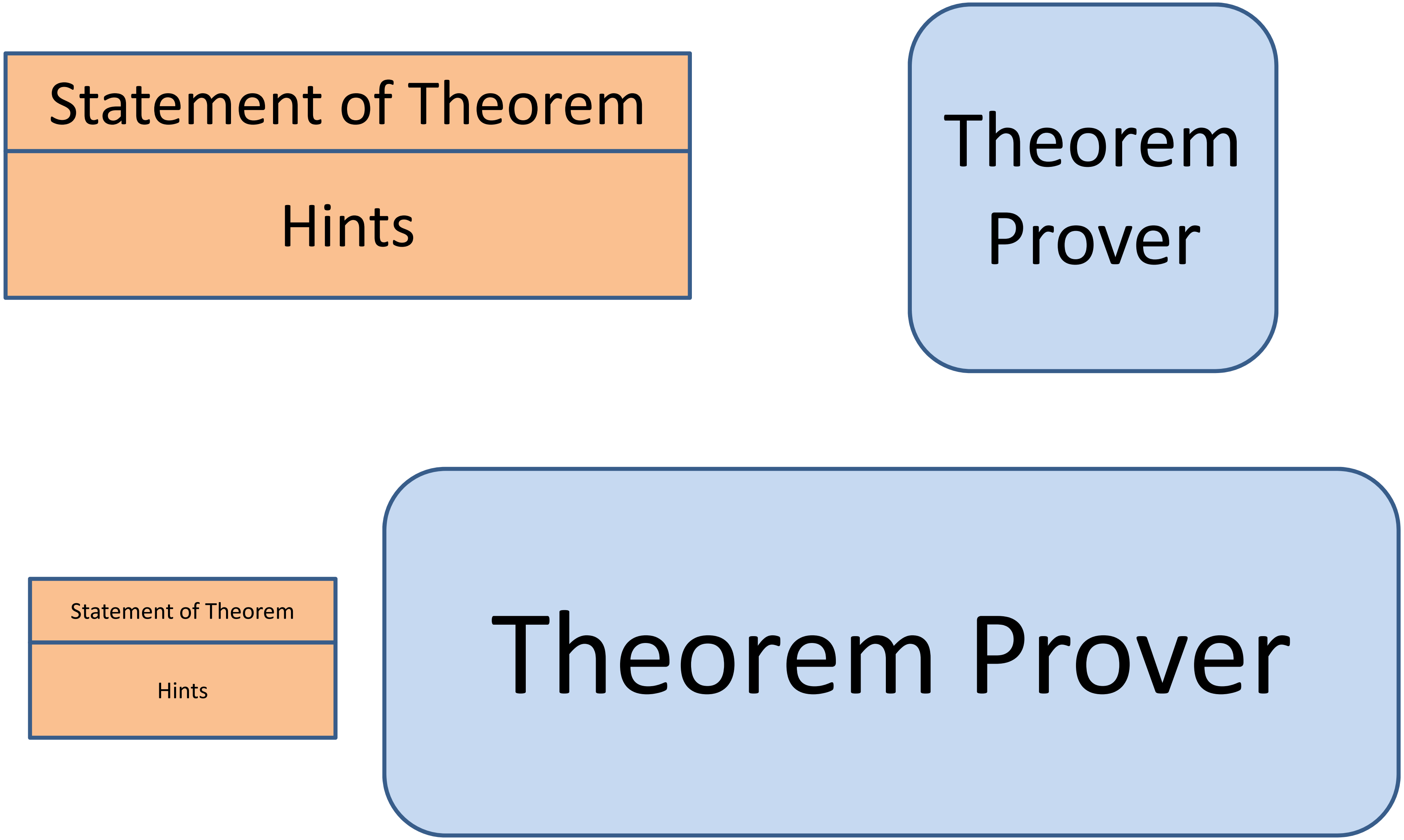
Theorem Prover Overview



Misleading Scales



Misleading Scales

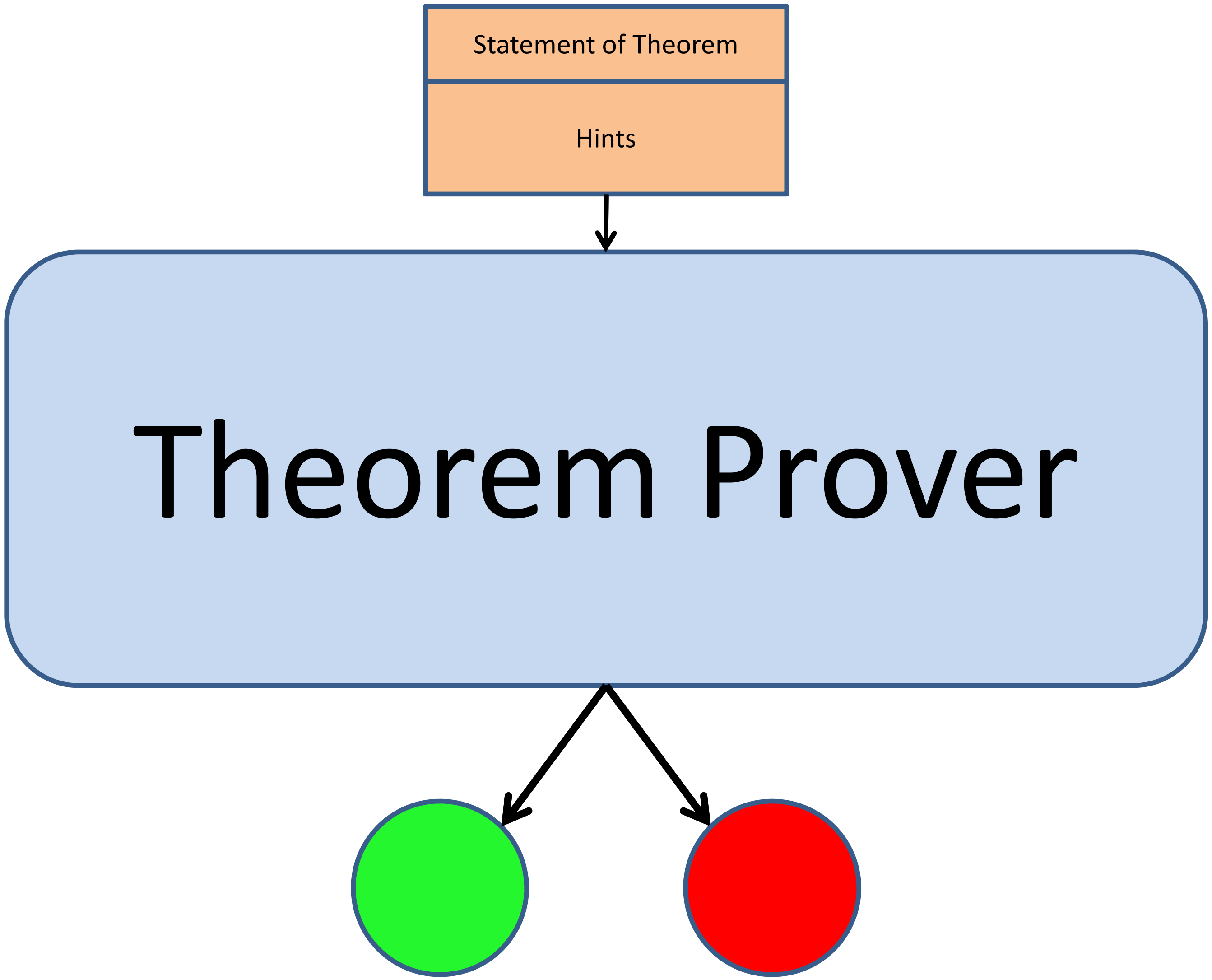


Trustworthiness

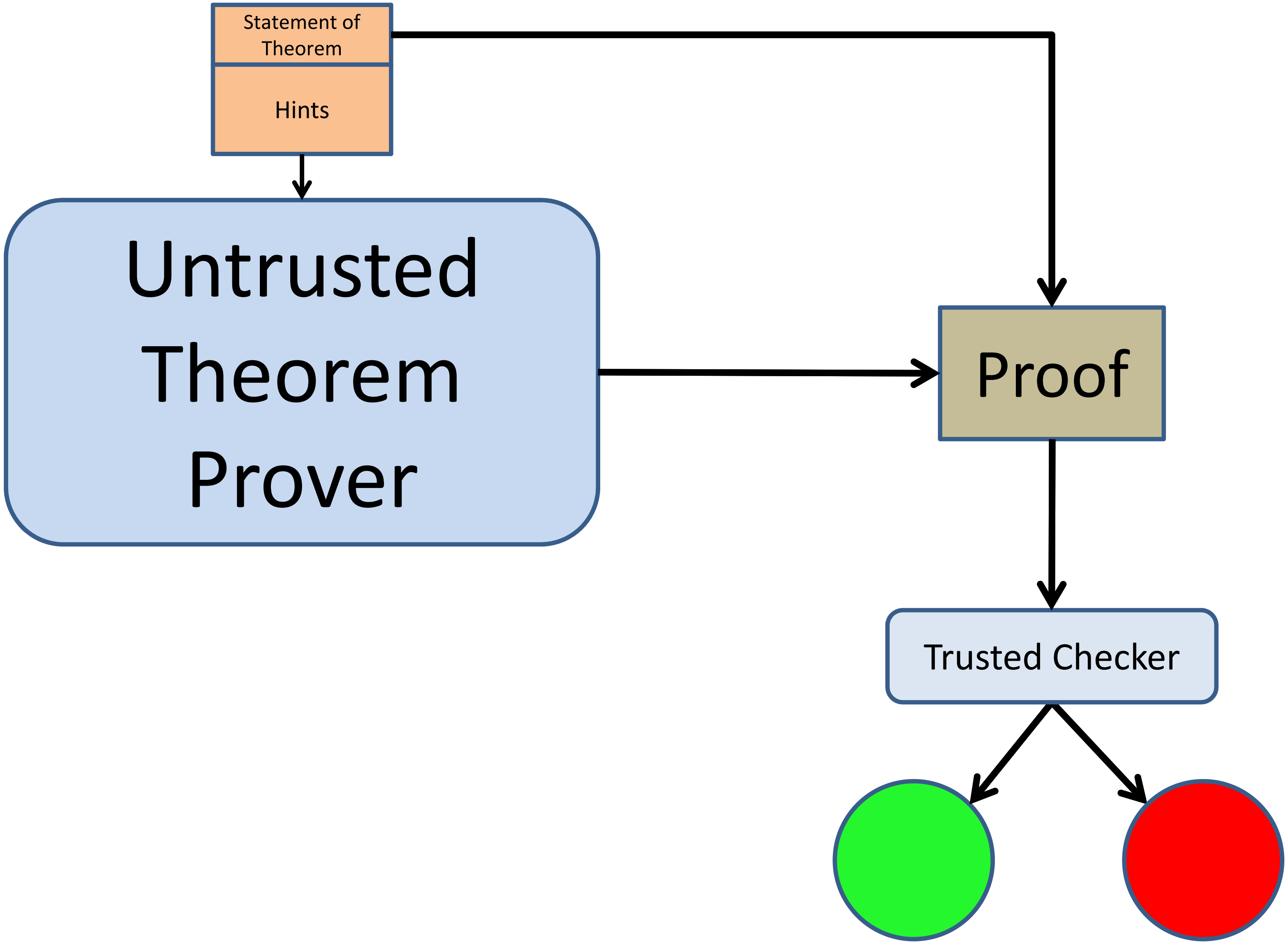
What kinds of things lead to increased trust?

- Complexity: simpler things better!
- Size: smaller things better!
- Stability: constant things better!
- Mechanically verified: much better!

Increasing Confidence



Increasing Confidence

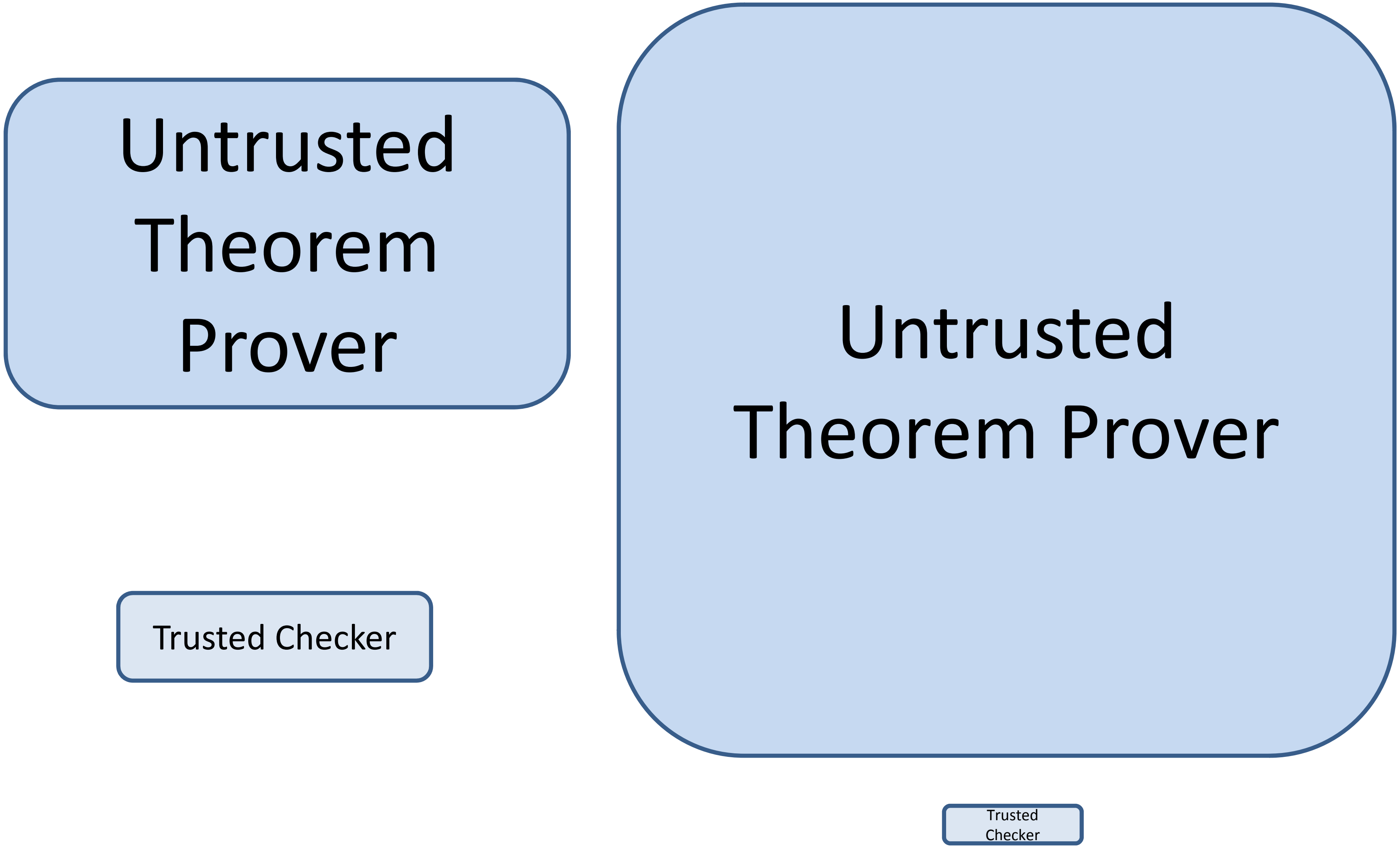


Misleading Scales

Untrusted Theorem Prover

Trusted Checker

Misleading Scales



Theorem Prover

- Generates proof from hints
- Frequently updated with new features
- Can be large (as large or larger than a compiler, 200k+ lines)
- Does not have to be trusted

Checker

- Checker is very:
  – Simple
  – Stable
  – Small
  – Verified by humans very carefully

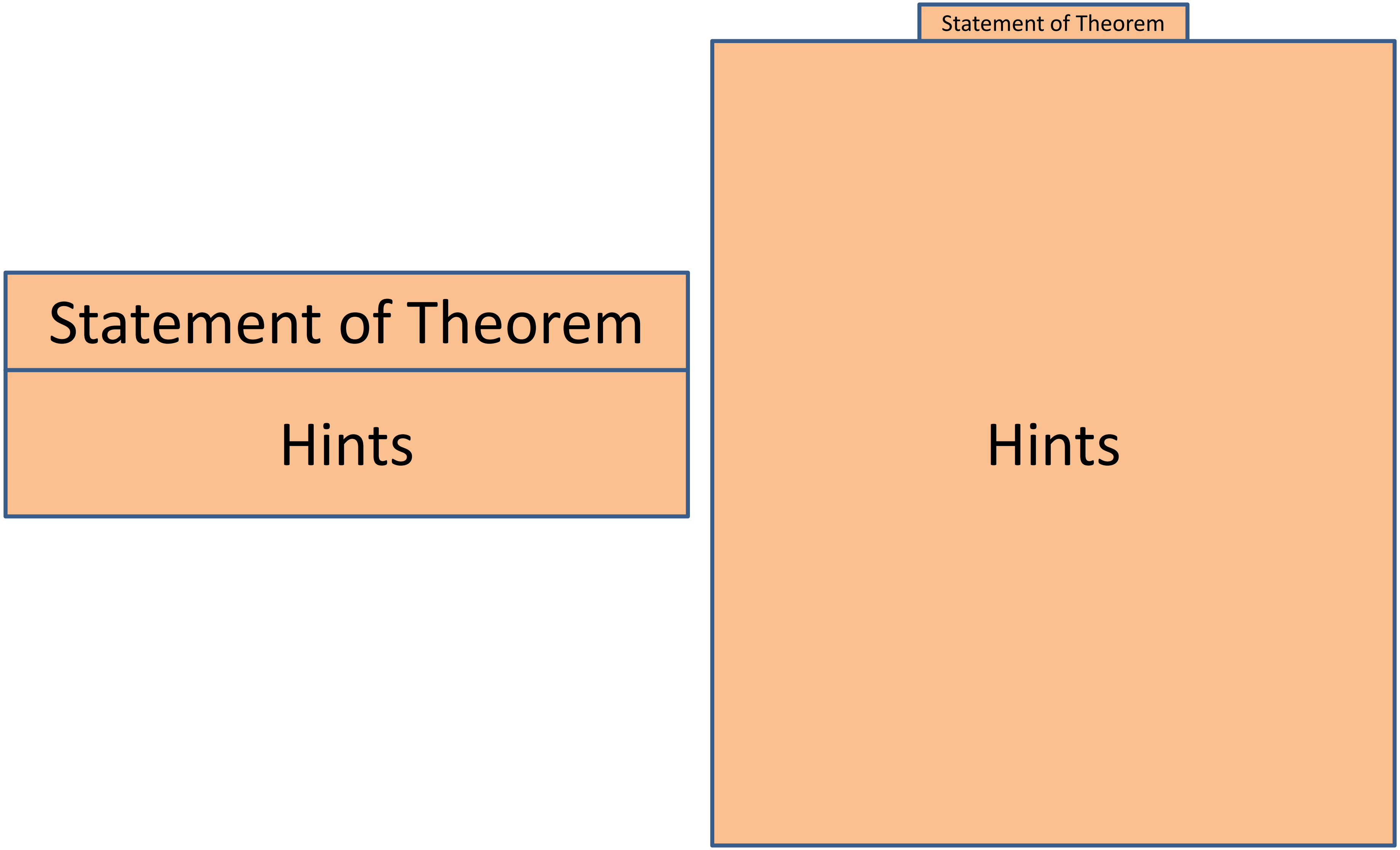
- Smallest known checker for HOL around 800 lines of C with no library support
  – Included parser and simple Prolog interpreter

Misleading Scales

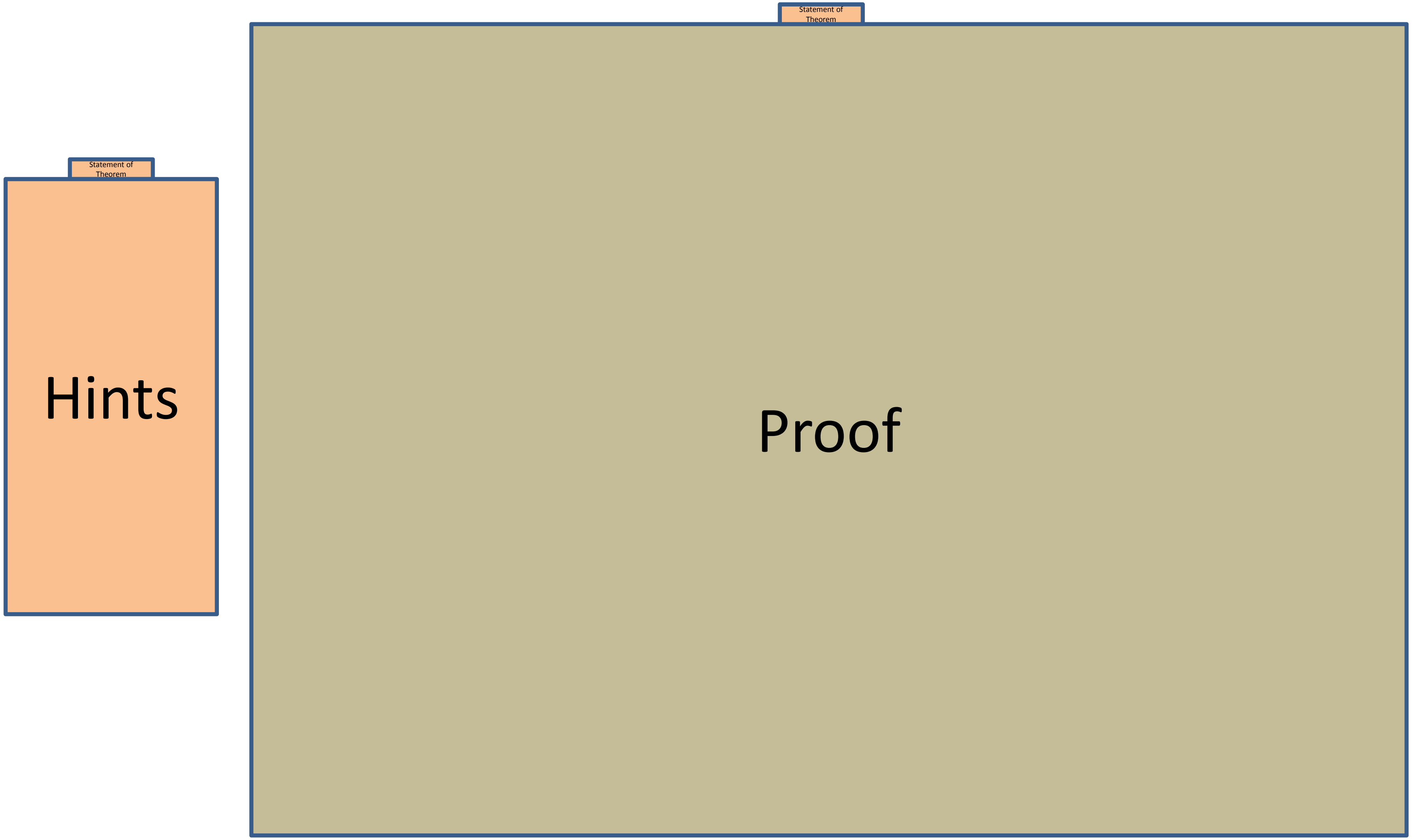
Statement of Theorem

Hints

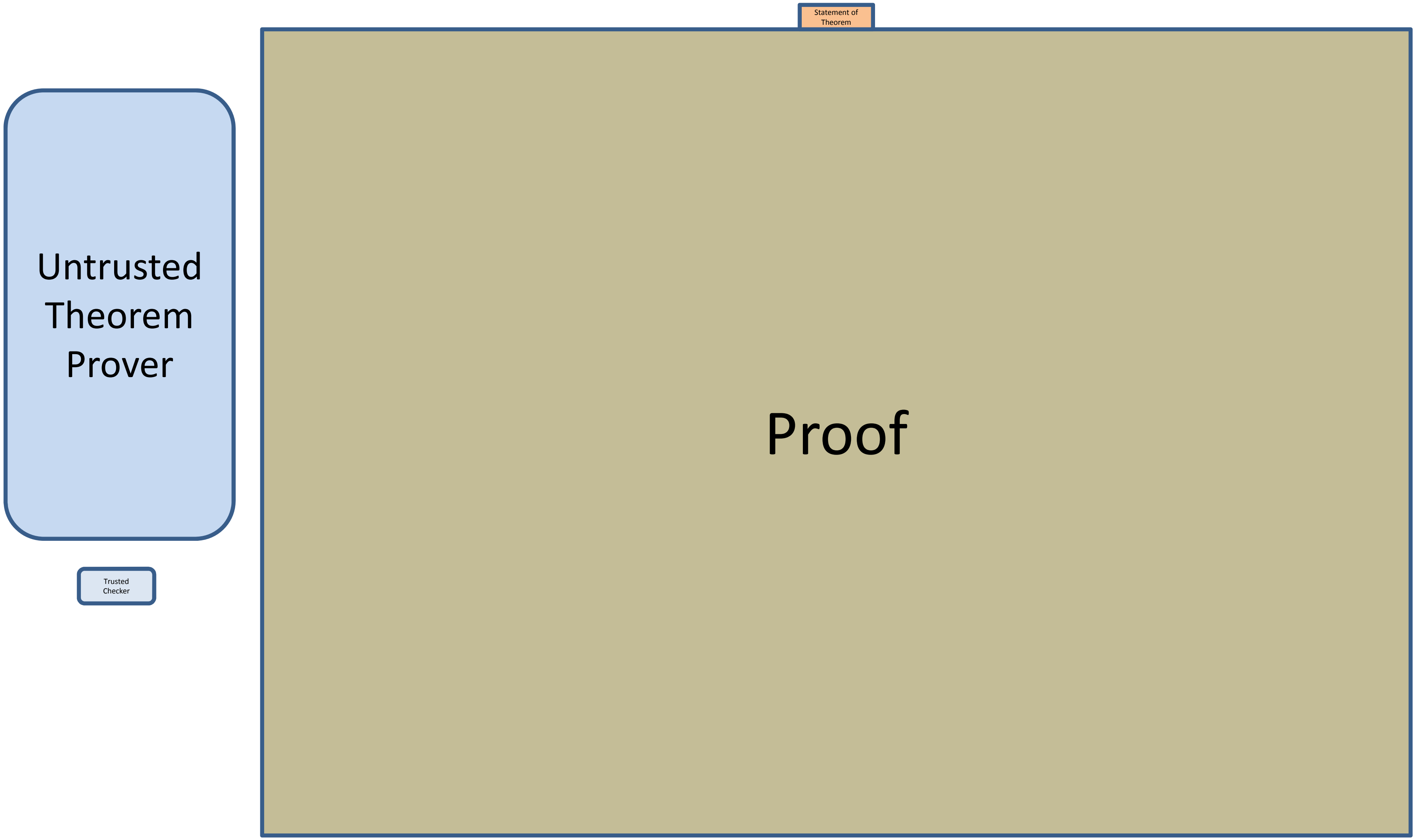
Misleading Scales



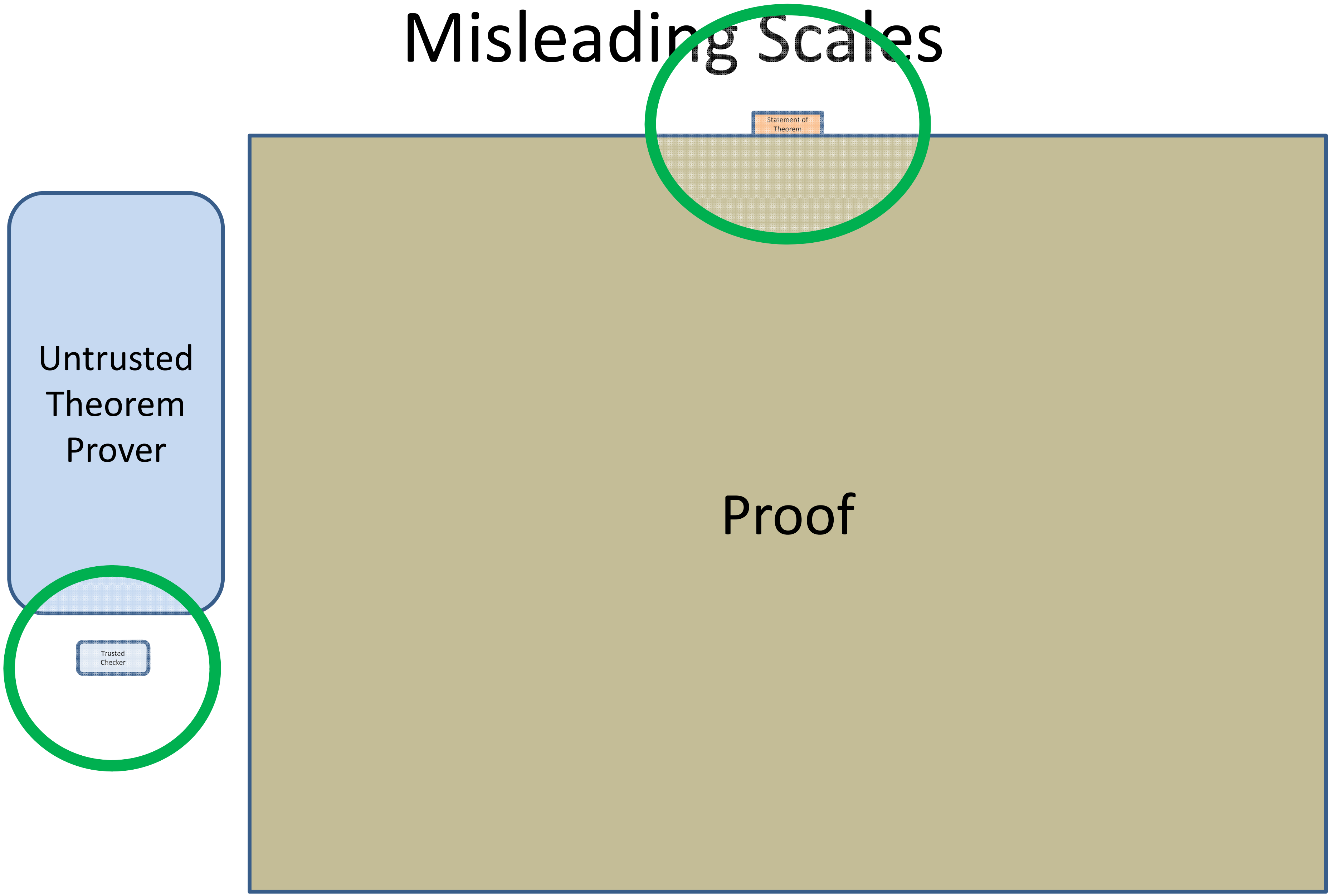
Misleading Scales



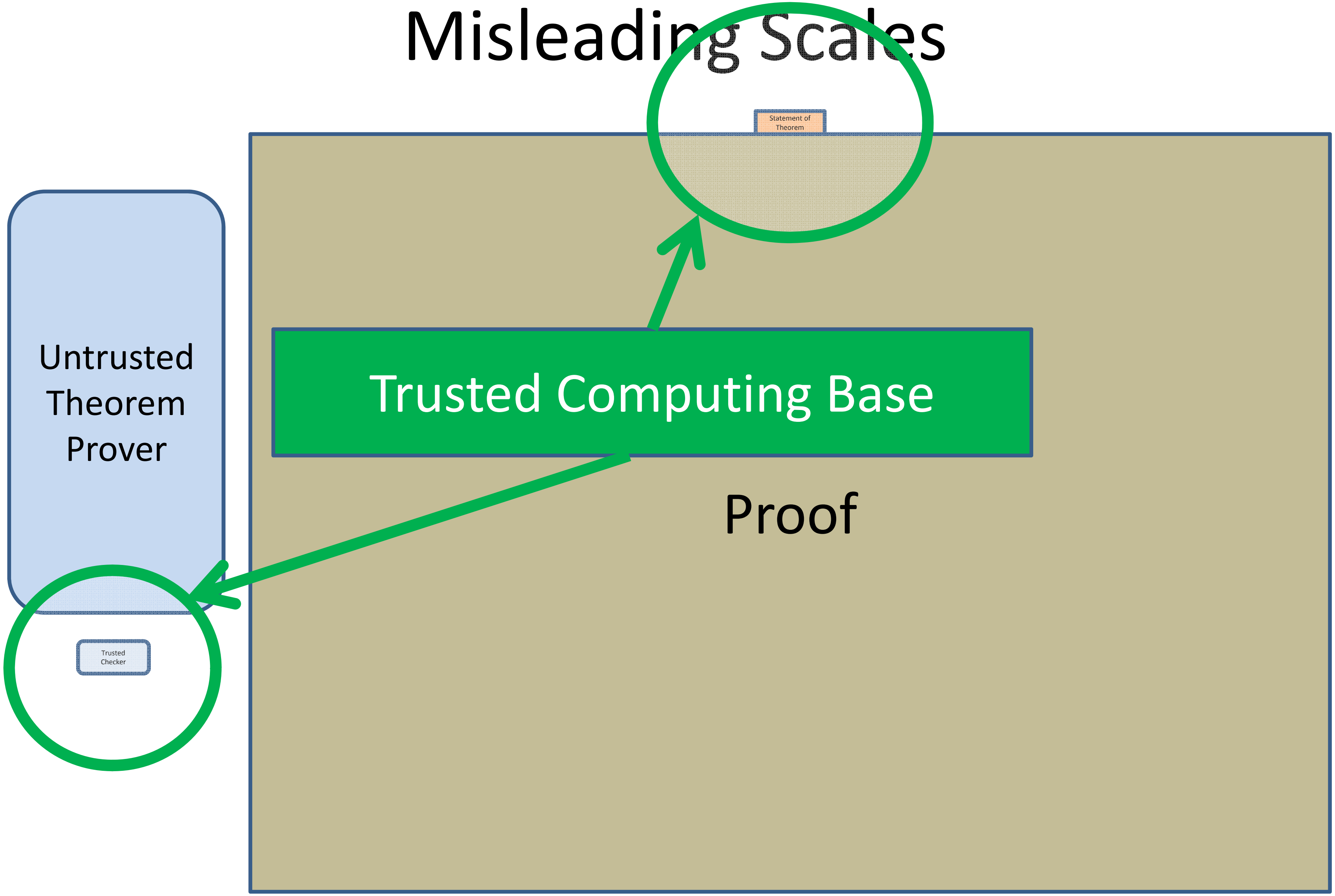
Misleading Scales



Misleading Scales



Misleading Scales



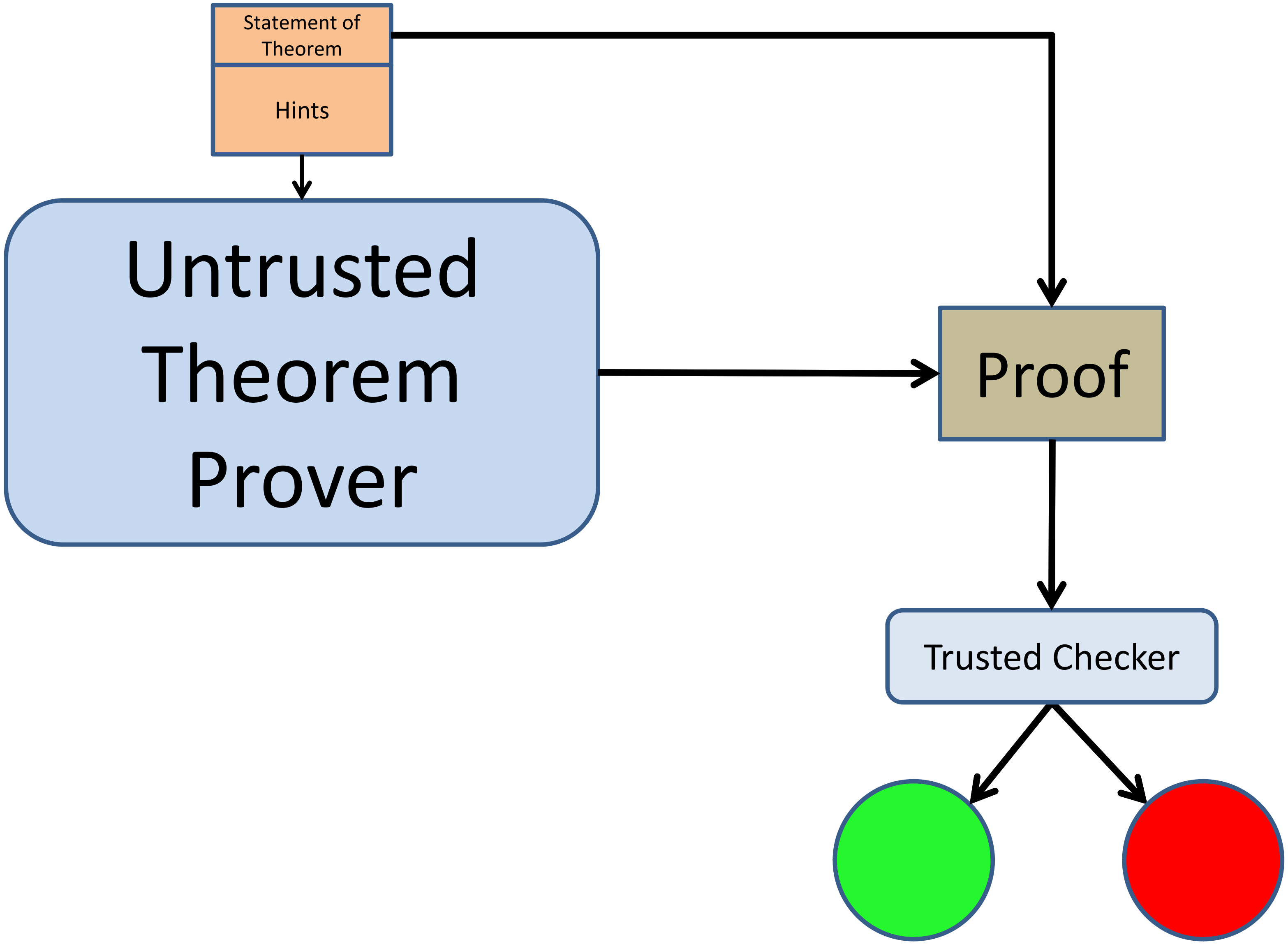
Trusted Computing Base

- The only things that have to be trusted:
  - Checker
  - Statement of theorem

- Everything else (hints, library, theorem prover, proof) does not

- Possible to get 3+ orders of magnitude difference in size (1000x) between trusted and untrusted

Review


Statement of Theorem
Hints
Untrusted Theorem Prover
Proof
Trusted Checker