CS 3110

Verification in Coq

Rachit Nigam
Spring 2019

Attendance question

Pick one of the following theorems. Then, a year from now, either you have to pay $10k or you get $10k.

- You pay if the theorem you picked turns out to have been discovered during that year to be demonstrably false.
- You get $10k otherwise.

A. A theorem you proved (and got full credit for) on a CS 2800 homework.

B. Chapter 2 of Prof. Foster's PhD dissertation.

C. The Coq theorem that the CompCert compiler correctly compiles the C programming language to x86.

D. The Pythagorean Theorem ($a^2 + b^2 = c^2$).

E. None of the above

Discussion: why???

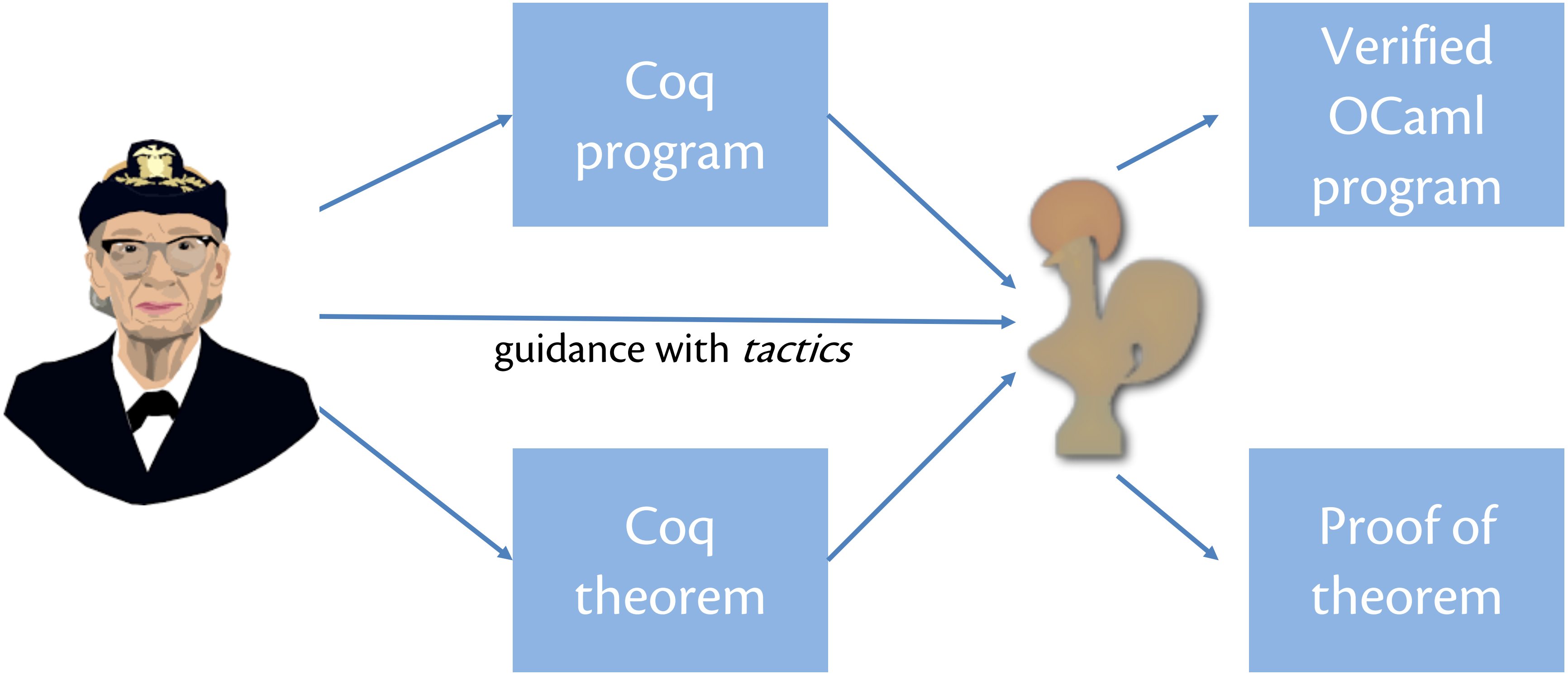
Review

Previously in 3110:

- Functional programming in Coq
- Logic in Coq
- Proofs are programs
- Induction in Coq

Today: Verification and extraction

Coq for program verification



Coq program

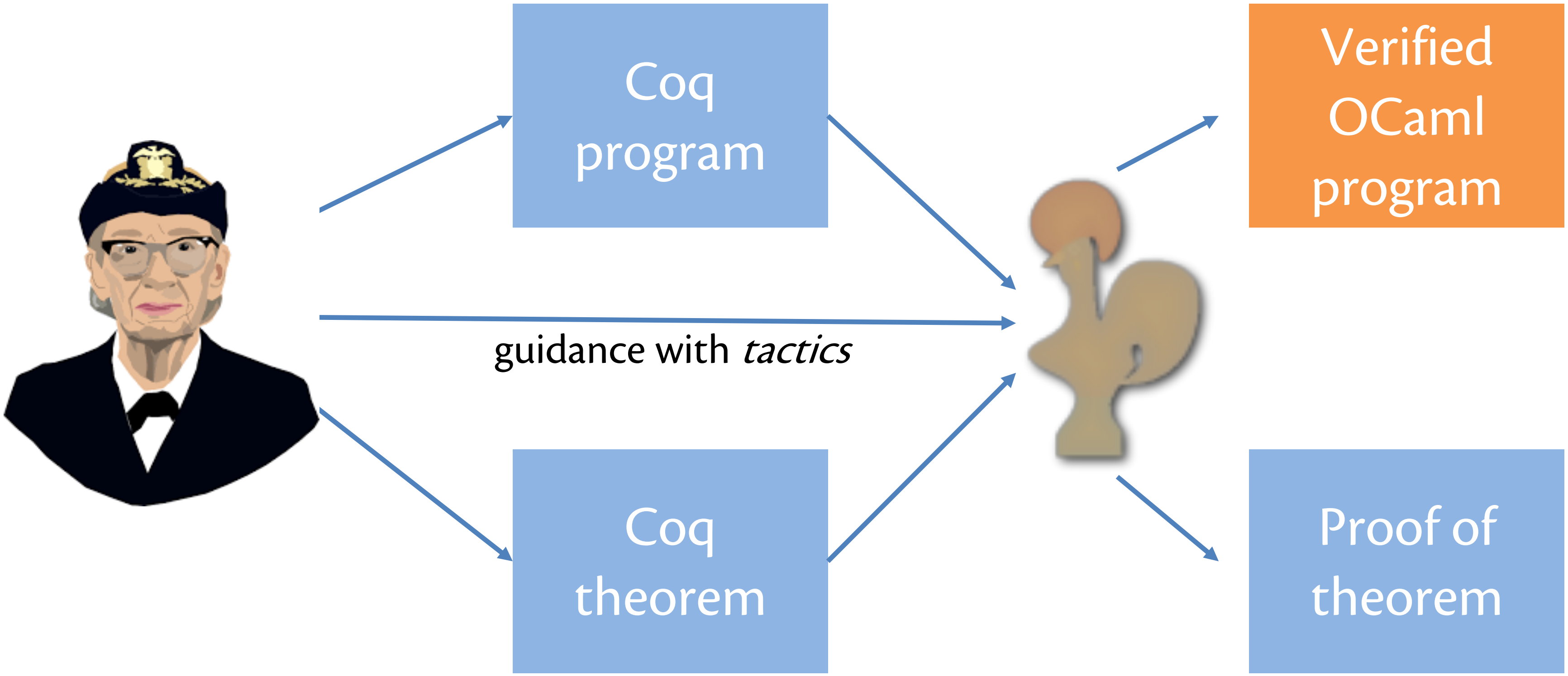
guidance with *tactics*

Coq theorem

Verified OCaml program

Proof of theorem

Coq for program verification



Coq program

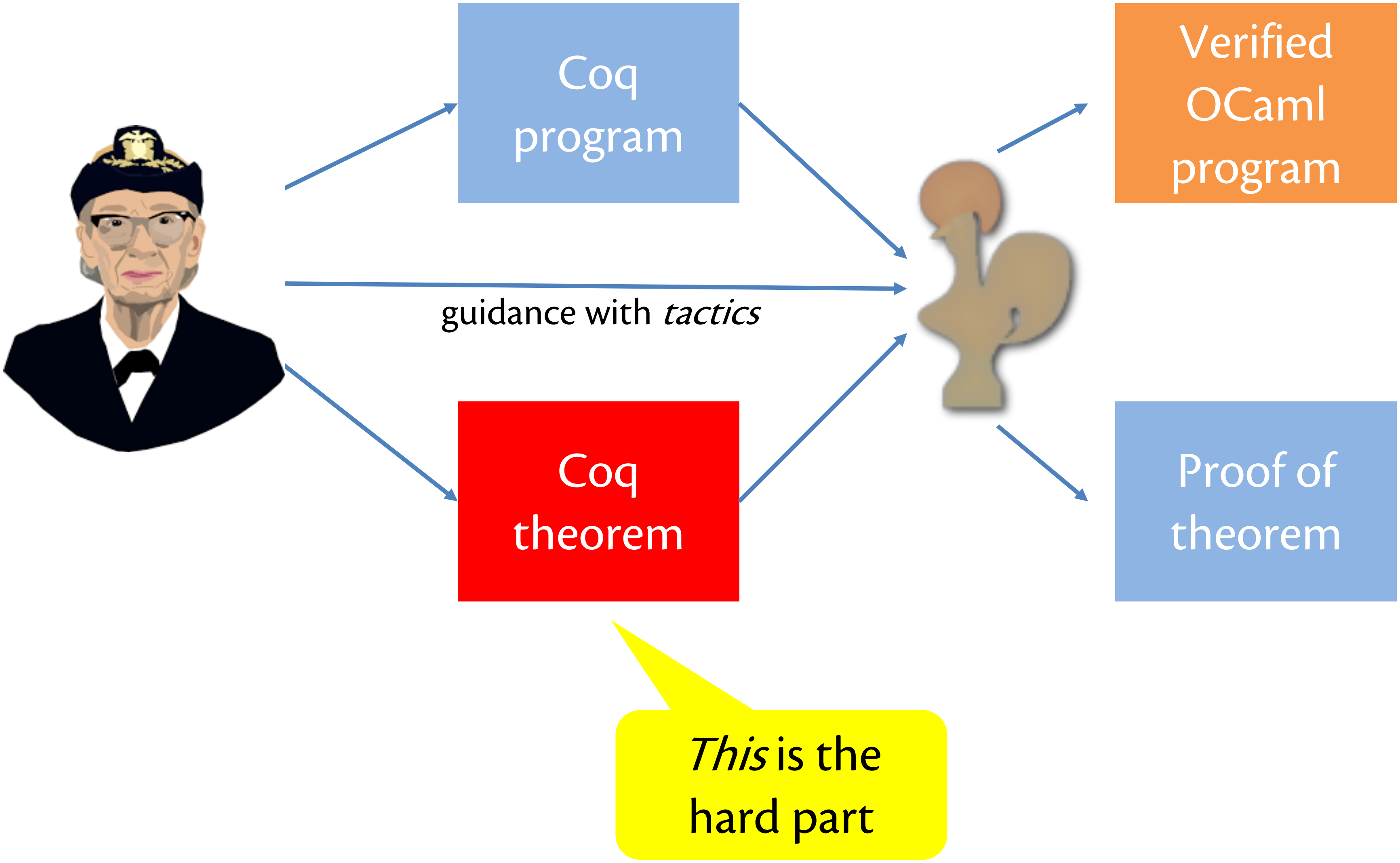
guidance with *tactics*

Coq theorem

Verified OCaml program

Proof of theorem

Coq for program verification



Coq program

guidance with *tactics*

Coq theorem

Verified OCaml program

Proof of theorem

This is the hard part

Theorems and test cases

- Do I have the right ones?
- Do I have enough?
- What am I missing?

… there are no great answers to these questions, only methodologies that help

ALGEBRAIC SPECIFICATION

Stack

```
module type Stack = sig
  type 'a t
  val empty    : 'a t
  val is_empty : 'a t -> bool
  val size     : 'a t -> int
  val peek     : 'a t -> 'a option
  val push     : 'a -> 'a t -> 'a t
  val pop      : 'a t -> 'a t option
end
```

Categories of operations

- **Creator:** creates value of type "from scratch" without any inputs of that type
- **Producer:** takes value of type as input and returns value of type as output
- **Observer:** takes value of type as input but does not return value of type as output

Stack

```
module type Stack = sig
  type 'a t
  val empty    : 'a t
  val is_empty : 'a t -> bool
  val size     : 'a t -> int
  val peek     : 'a t -> 'a option
  val push     : 'a -> 'a t -> 'a t
  val pop      : 'a t -> 'a t option
end
```

creator

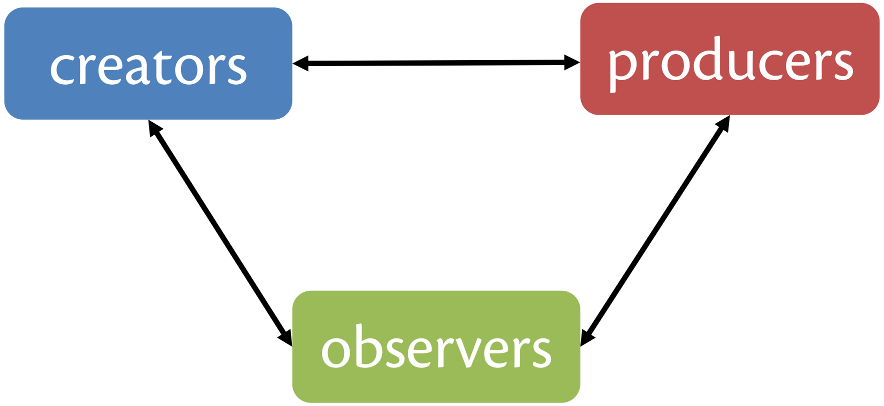
observers

producers

Algebraic specification

aka *equational specification*

```
is_empty empty = true
```



Stack

```
module type Stack = sig
  type 'a t
  val empty    : 'a t
  val is_empty : 'a t -> bool
  val size     : 'a t -> int
  val peek     : 'a t -> 'a option
  val push     : 'a -> 'a t -> 'a t
  val pop      : 'a t -> 'a t option
end
```

Discussion: invent equational specification for stacks

Stack specification

- `is_empty empty = true`
- `is_empty (push _ _) = false`
- `peek empty = None`
- `peek (push x _) = Some x`
- `size empty = 0`
- `size (push _ s) = 1 + size s`
- `pop empty = None`
- `pop (push _ s) = Some s`

VERIFICATION AND EXTRACTION

Demo

SPECIFICATION WITH INDUCTIVE PROPOSITIONS

Factorial

- **Precondition:** `n >= 0`
- **Postcondition:** `fact n = n!`

- **Problem:** how to express `!` in Coq?

Demo

Specifying factorial as a relation

$$\frac{}{\text{factorial\_of}(0, 1)}$$

Axiom: what is factorial of zero?

$$\frac{\text{factorial\_of}(a, b)}{\text{factorial\_of}(a+1, (a+1)*b)}$$

Inference rule: what is factorial of successor?

Demo

DEPARTMENT OF

REDUNDANCY

DEPARTMENT

SPECIFICATION WITH REFERENCE IMPLEMENTATIONS

Demo

Upcoming events

[Today] Foster out of town, no Office Hours

[Today] A9 released (it will be fun, short)

[Friday] A8 due

This is verified.

THIS IS 3110