Verification in Coq

Prof. Clarkson
Fall 2018

Today's music: *Check Yo Self* by Ice Cube
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. A lemma in the appendix of Prof. Clarkson’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

Coq theorem

guidance with tactics

Verified OCaml program

Proof of theorem
Coq for program verification

- Coq program
- Coq theorem
- Guidance with tactics
- Verified OCaml program
- Proof of theorem
Coq for program verification

Coq program

guidance with tactics

Verified OCaml program

Coq theorem

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
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
Algebraic specification

aka *equational specification*

\[ \text{is\_empty} \quad \text{empty} \quad = \quad \text{true} \]
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
SPECIFICATION WITH INDUCTIVE PROPOSITIONS
Factorial

- **Precondition**: \( n \geq 0 \)
- **Postcondition**: `fact n = n!`

- **Problem**: how to express `!` in Coq?
Specifying factorial as a relation

Axiom: what is factorial of zero?

\[
\text{factorial\_of}(0, 1)
\]

Inference rule: what is factorial of successor?

\[
\text{factorial\_of}(a, b) \\
\text{factorial\_of}(a+1, (a+1)\times b)
\]
DEPARTMENT OF REDUNDANCY DEPARTMENT
SPECIFICATION WITH REFERENCE IMPLEMENTATIONS
Upcoming events

Thanksgiving Break!

This is verified.

THIS IS 3110