Program Verification using Coq

Daniel Britten

September 21, 2017

Example: Plus

```
Fixpoint plus (n : nat) (m : nat) : nat :=
  match n with
    | O ⇒ m
    | S n' ⇒ S (plus n' m)
  end.
```

```
Theorem plus_n_O : ∀ n:nat, n = n + 0.
Proof.
  intros n. induction n as [| n' IHn'].
  - reflexivity.
  - simpl. rewrite ← IHn'. reflexivity. Qed.
```

Source: Software Foundations Textbook.

Example: Even numbers - 'How'

```
Fixpoint evenb (n : nat) : bool :=
match n with
| 0 ⇒ true
| 1 ⇒ false
| S (S n) ⇒ evenb n
end.
```

Example: Even numbers - 'What'

$\texttt{Definition}$ Even $(n : \mathit{nat}) : \texttt{Prop} :=$
$\exists\ m,\ n = 2 \times m.$

Example: Even numbers - 'Correctness'

`Lemma` $evenb\_correct : \forall\ n,\ evenb\ n = true \leftrightarrow Even\ n.$

Example: Even numbers - 'Why'

Lemma *lemma_one* : $\forall$ *n*, *evenb* (*S* *n*) = *negb* (*evenb* *n*).
Proof.
induction *n* as [|*n'*].
- Base case: $n = 0$
  reflexivity.
- Inductive case: for some n', $n = S\ n'$
  rewrite *IHn'*. simpl.
  destruct (*evenb* *n'*) *eqn*:*SubCase*.
  + Sub-case (*evenb* *n'*) = *true*
    reflexivity.
  + Sub-case (*evenb* *n'*) = *false*
    reflexivity.

Qed.

Example: Even numbers - 'Why'

Lemma *lemma_two* : $\forall$ *n*, *Even* (*S n*) $\leftrightarrow \neg$ *Even n*.

Proof.

split.

- induction *n* as [|*n'*].
  - Base case: $n = 0$: unfold *Even*. intros. destruct *H*. destruct *x*. inversion *H*. simpl in *H*. rewrite $\leftarrow$ *plus_n_Sm* in *H*. discriminate *H*.
  - Inductive case: for some *n'*, $n = S\ n'$: intros. unfold *not*. intros. apply *IHn'*. exact *H0*. unfold *Even*. destruct *H*. destruct *x*. simpl in *H*. discriminate *H*. $\exists$ *x*. simpl in *H*. rewrite $\leftarrow$ *plus_n_Sm* in *H*. inversion *H*. simpl. reflexivity.
- induction *n*. intros. unfold *Even* in *H*. *exfalso*. apply *H*. $\exists$ 0. reflexivity.

intros. apply *NNPP*. unfold *not*. intros. apply *H*. apply *IHn*. unfold *not*. intros. apply *H0*. unfold *Even* in *H*. destruct *H1*. unfold *Even*. $\exists$ (*S x*). rewrite *H1*. simpl. rewrite $\leftarrow$ *plus_n_Sm*. reflexivity. Qed.

Example: Even numbers - 'Why'

Lemma *evenb_correct* : $\forall$ *n*, *evenb n* = *true* $\leftrightarrow$ *Even n*.

Proof.

induction *n*.

- firstorder. unfold *Even*. $\exists$ 0. reflexivity.
- split.
  + intros. rewrite $\rightarrow$ *lemma_one* in *H*. rewrite *lemma_two*. unfold *not*. intros. apply *IHn* in *H0*. rewrite *H0* in *H*. simpl in *H*. discriminate *H*.
  + intros. apply *lemma_two* in *H*. rewrite *lemma_one*. destruct (*evenb n*) *eqn*:*Case*.
    × *exfalso*. apply *H*. apply *IHn*. reflexivity.
    × reflexivity.

Qed.

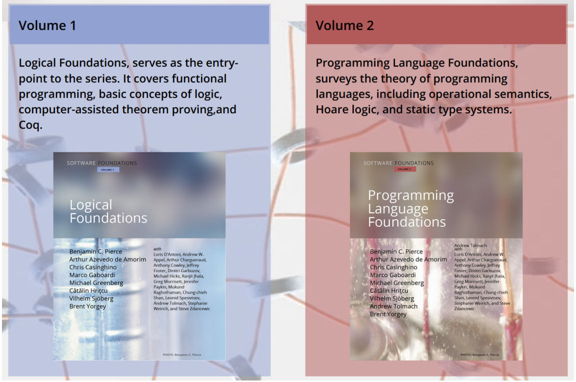
Example: Even numbers - 'What+How'

```
Program Definition evenb'
    (n : nat) : {b:bool | b = true ↔ Even n} :=
match n with
| 0 ⇒ true
| 1 ⇒ false
| S (S n) ⇒ evenb n
end.
```

Example: Even numbers - 'Why'

```
Next Obligation.
firstorder. ∃ 0. reflexivity. Qed.
Next Obligation.
firstorder. inversion H. destruct x. inversion H. simpl in
H. rewrite ← plus_n_Sm in H. inversion H. Qed.
Next Obligation.
induction n.
 - firstorder. unfold Even. ∃ 1. reflexivity.
 - split.
    + intros. rewrite → lemma_one in H. rewrite
lemma_two. unfold not. intros. apply IHn in H0. rewrite H0
in H. simpl in H. discriminate H.
    + intros. apply lemma_two in H. rewrite lemma_one.
destruct (evenb n) eqn:Case.
          × exfalso. apply H. apply IHn. reflexivity.
          × reflexivity.
Qed.
```

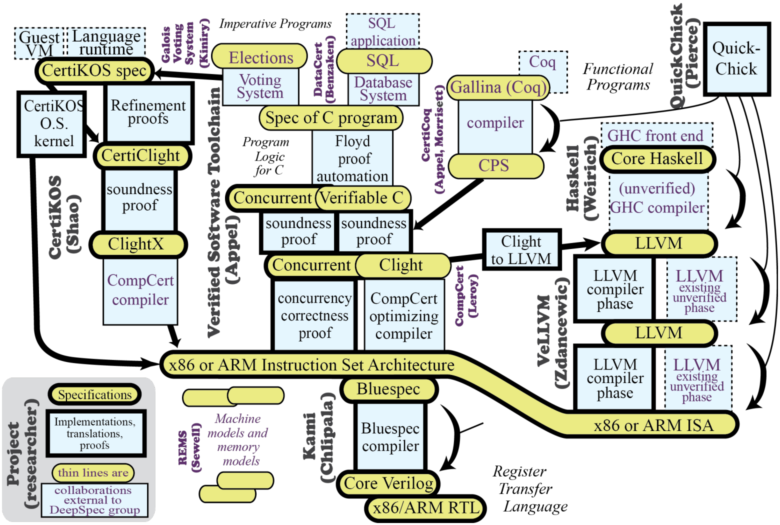
Software Foundations



https://softwarefoundations.cis.upenn.edu/

- VellVM
- CompCert
- Spark Ada (e.g. Air traffic management systems)
- Model Checking

DeepSpec


Imperative Programs
SQL application
Guest VM
Language runtime
Galois Voting System (Kiniry)
Elections
DataCert (Benzaken)
SQL
Coq
Functional Programs
QuickChick (Pierce)
Quick-Chick
CertiKOS spec
Voting System
Database System
Gallina (Coq)
CertiKOS O.S. kernel
Refinement proofs
Spec of C program
compiler
CertiClight
Program Logic for C
Floyd proof automation
CertiCoq (Appel, Morrisett)
CPS
GHC front end
Core Haskell
Haskell (Weirich)
CertiKOS (Shao)
soundness proof
Verified Software Toolchain (Appel)
Concurrent
Verifiable C
(unverified) GHC compiler
soundness proof
soundness proof
ClightX
Clight to LLVM
LLVM
Concurrent
Clight
CompCert compiler
CompCert (Leroy)
LLVM compiler phase
LLVM existing unverified phase
concurrency correctness proof
CompCert optimizing compiler
VeLLVM (Zdancewic)
LLVM
x86 or ARM Instruction Set Architecture
LLVM compiler phase
LLVM existing unverified phase
Project (researcher)
Specifications
Implementations, translations, proofs
thin lines are
collaborations external to DeepSpec group
Bluespec
Kami (Chlipala)
REMS (Sewell)
Machine models and memory models
Bluespec compiler
x86 or ARM ISA
Core Verilog
Register Transfer Language
x86/ARM RTL