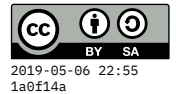


Formal Methods



[Rough notes]

Treating software, or specifications of software, as *mathematical* objects, so we can prove theorems about them.

- Model checking specifications with TLA+ Toolbox. TLA = Temporal Logic of Actions. [Leslie Lamport] – Sample specification of a bank account transfer
- Program logics in a proof assistant (Coq) [my example below].

```
Inductive nat : Type :=
| Z : nat
| S : nat -> nat.
```

```
Inductive even : nat -> Prop :=
| evZ : even Z
| evS : forall n:nat,
    even n -> even (S(S n)).
```

```
Theorem six_even:
  even(S(S(S(S(S(S Z)))))).
```

```
Proof.
  apply evS.
  apply evS.
  apply evS.
  apply evZ.
```

Qed.

```
Fixpoint plus (n:nat) (m:nat): nat :=
  match n with
  | Z => m
  | S k => S (plus k m)
  end.
```

```
Eval compute in plus (S(S Z)) (S(S(S Z))).
```

```
Theorem ev_plus_ev_is_ev:
  forall n m : nat,
    even n ->
    even m ->
    even (plus n m).
```

Proof.

```
intros n m EN EM.  
induction EN as [|n2 EN2].  
simpl. exact EM.  
simpl. apply evS. exact IHEN2.
```

Qed.

Inductive odd : nat -> Prop :=

```
| odd1: odd (S Z)  
| oddS: forall n:nat, odd n -> odd(S(S n)).
```

Lemma succ_odd_is_even:

```
forall n:nat, odd n -> even (S n).
```

Proof.

```
intros n OddN.  
induction OddN.  
apply evS. apply evZ.  
apply evS. exact IHOddN.
```

Qed.