Manifest Termination

Assia Mahboubi Guillaume Melquiond

Inria

June 15, 2023

Mathematics, Proofs, and Computations

Helfgott's proof of the ternary Goldbach conjecture

Every odd number greater than 5 is the sum of three primes.

(250+ pages of proofs, 1000+ hours of computations)

Mathematics, Proofs, and Computations

Helfgott's proof of the ternary Goldbach conjecture

Every odd number greater than 5 is the sum of three primes.

(250+ pages of proofs, 1000+ hours of computations)

Long-term goal: get mathematicians to do formal proofs

Intermediate steps:

- get them to use a proof assistant,
  - let them perform computations inside a proof assistant,
    - let them write their programs inside a proof assistant.

Mathematics, Proofs, and Computations

Helfgott's proof of the ternary Goldbach conjecture

Every odd number greater than 5 is the sum of three primes.

(250+ pages of proofs, 1000+ hours of computations)

Long-term goal: get mathematicians to do formal proofs

Intermediate steps:

- get them to use a proof assistant,
  - let them perform computations inside a proof assistant,
    - let them write their programs inside a proof assistant.

What about non-termination?

If an evaluation actually terminates,
why do we still need to prove that it terminates?

Example: Factorial

Factorial over $\mathbb{N} = 1 + \mathbb{N}$

```
Fixpoint fact (n: nat): nat :=
  match n with
  | O => 1
  | S n' => n * fact n'
  end.
```

Termination

Reason: recursive calls on a structurally decreasing argument.

Example: Factorial

Factorial over $\mathbb{N} = 1 + \mathbb{N}$

```
Fixpoint fact (n: nat): nat :=
  match n with
  | O => 1
  | S n' => n * fact n'
  end.
```

Termination

Reason: recursive calls on a structurally decreasing argument.

Computations

```
Goal fact 15 = 1307674368000.
Proof. reflexivity. Qed.
```

Does not terminate in practice!

Factorial, Faster

Factorial over $\mathbb{Z} = 1 + 2\mathbb{P}$, $\mathbb{P} = 1 + 2\mathbb{P}$

```
Fixpoint factZ (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (factZ (Z.pred n)).
```

No longer structurally decreasing. (Not even terminating in general.)
But `factZ 15` would terminate!

Factorial, Faster

Factorial over $\mathbb{Z} = 1 + 2\mathbb{P}$, $\mathbb{P} = 1 + 2\mathbb{P}$

```
Fixpoint factZ (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (factZ (Z.pred n)).
```

No longer structurally decreasing. (Not even terminating in general.)
But `factZ 15` would terminate!

Usual solution: provide some structurally decreasing fuel

- Variant 1: prove beforehand that there is enough fuel,
  e.g., provide a well-founded relation and an accessibility proof.
- Variant 2: deal with a possibly stuck computation,
  e.g., use the option monad.

Factorial, the Hard Way

```
Lemma fact2_aux1 n: (0 <= n)%Z -> Z.eqb n 0 = false ->
     (0 <= Z.pred n)%Z.
Proof. case Z.eqb_spec ; [easy|lia]. Qed.
Lemma fact2_aux2 n: (0 <= n)%Z -> Zwf 0 (Z.pred n) n.
Proof. unfold Zwf. lia. Qed.

Fixpoint factZ_aux (n: Z) (H: Z.le 0 n)
                   (W: Acc (Zwf 0) n): Z :=
  ( if Z.eqb n 0 as b return Z.eqb n 0 = b -> Z
    then fun _ => 1%Z
    else fun Hn =>
      match W with
      | Acc_intro _ W' =>
        Z.mul n (factZ_aux (Z.pred n)
          (fact2_aux1 n H Hn) (W' _ (fact2_aux2 n H)))
      end
  ) eq_refl.

Definition factZ n (H: Z.le 0 n) :=
  factZ_aux n H (Acc_intro_generator 20 (
     Zwf_well_founded 0) n).

Eval compute in factZ 15 _.
```

Open Recursion and Computations

Open factorial over $\mathbb{Z} = 1 + 2\mathbb{P}$, $\mathbb{P} = 1 + 2\mathbb{P}$

```
Definition factZ (k: Z -> Z) (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (k (Z.pred n)).
```

`factZ` has type $(\mathbb{Z} \to \mathbb{Z}) \to (\mathbb{Z} \to \mathbb{Z})$.

(More generally, $F : (T \to U) \to (T \to U)$.)

We want to compute "$F^* x$", i.e., "$(F \circ \cdots \circ F \circ \ldots)\, x$".

Open Recursion and Computations

Open factorial over $\mathbb{Z} = 1 + 2\mathbb{P}$, $\mathbb{P} = 1 + 2\mathbb{P}$

```
Definition factZ (k: Z -> Z) (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (k (Z.pred n)).
```

`factZ` has type $(\mathbb{Z} \to \mathbb{Z}) \to (\mathbb{Z} \to \mathbb{Z})$.

(More generally, $F : (T \to U) \to (T \to U)$.)

We want to compute "$F^* x$", i.e., "$(F \circ \cdots \circ F \circ \dots) x$".

Bounded computations

```
Goal forall k, iter 100 factZ k 15 = 1307674368000%Z.
Proof. reflexivity. Qed.
```

Next Step: Proving Something Interesting

Knowing "$\forall k,\ F^n\, k\, x = y$", can we prove anything about y?

Example: $(0 \leq x \Rightarrow y = x!)$ for `factZ`.

Next Step: Proving Something Interesting

Knowing "$\forall k,\ F^n\, k\, x = y$", can we prove anything about y?
Example: $(0 \le x \Rightarrow y = x!)$ for `factZ`.

```
Lemma nat_spec {T U} (F:(T->U)->T->U) (P:T->U->Prop):
  (∃ f, ∀ x, P x (f x)) ->                           (*1*)
  (∀ k x, (∀ y, P y (k y)) -> P x (F k x)) ->        (*2*)
  ∀ n x y, (∀ h, Nat.iter n F h x = y) ->            (*3*)
  P x y.
```

Requirements

1. Some function satisfies the specification P.
2. P is an invariant of F.
3. The computation $F^*\, x$ terminates.

Back to factorial

```
Definition factZ (k: Z -> Z) (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (k (Z.pred n)).

Goal Z.of_nat (fact 15) = 1307674368000%Z.
Proof.
apply nat_spec with (F := factZ) (n := 20)
  (P x y := Z.le 0 x -> Z.of_nat(fact(Z.to_nat x))=y).
- eexists; reflexivity.
- ...   (* Z.of_nat is a morphism *)
- reflexivity.
Qed. (* axiom-free *)
```

Manifest Termination

Back to factorial

```
Definition factZ (k: Z -> Z) (n: Z): Z :=
  if Z.eqb n 0 then 1%Z
  else Z.mul n (k (Z.pred n)).

Goal Z.of_nat (fact 15) = 1307674368000%Z.
Proof.
apply nat_spec with (F := factZ) (n := 20)
  (P x y := Z.le 0 x -> Z.of_nat(fact(Z.to_nat x))=y).
- eexists; reflexivity.
- ...   (* Z.of_nat is a morphism *)
- reflexivity.
Qed. (* axiom-free *)
```

It works just fine

But can we do without the hypothesis "$\exists f,\ \forall x,\ P\,x\,(f\,x)$"?

Going Further

A Constructive Fixed Point Combinator (Charguéraud 09)

```
Definition fixacc {T U} (dummy: U) (F: (T->U)->T->U)
    (R: T->T->bool) (x: T) (W: Acc R x) :=
  Acc_rect _ (fun u _ k =>
    let k' v := if R v u then k v _ else dummy in
    F k' u) W.
```

Just a fancy way of writing "$F^* x$"

- "$R\, v\, u$" describes the call graph. It holds when
  - either $F\, u$ was never called during the evaluation of $F^* x$,
  - or $F\, u$ performed a direct call to $F\, v$.
- W proves that the graph rooted at x is finite, acyclic.

Axioms! Lots of Axioms!

Assume the existence of a symbol

```
fixrel: ((T -> U) -> (T -> U)) -> T -> U -> Prop
```

such that `(fixrel F x y)` holds when $F^* x \equiv y$.

Axioms! Lots of Axioms!

Assume the existence of a symbol

```
fixrel: ((T -> U) -> (T -> U)) -> T -> U -> Prop
```

such that `(fixrel F x y)` holds when $F^* x \equiv y$.

```
Axiom fixrel_spec: ∀ {T U} (F:(T->U)->T->U) x y,
  fixrel F x y ->
  ∃ R,
  (∀ u f1 f2, (∀ v, R v u -> f1 v = f2 v) ->
              F f1 u = F f2 u) /\
  ∃ W: Acc R x,
  ∀ d, y = fixacc d F R x W.
```

Axioms! Lots of Axioms!

Assume the existence of a symbol

```
fixrel: ((T -> U) -> (T -> U)) -> T -> U -> Prop
```

such that `(fixrel F x y)` holds when $F^* \, x \equiv y$.

```
Axiom fixrel_spec: ∀ {T U} (F:(T->U)->T->U) x y,
  fixrel F x y ->
  ∃ R,
  (∀ u f1 f2, (∀ v, R v u -> f1 v = f2 v) ->
              F f1 u = F f2 u) /\
  ∃ W: Acc R x,
  ∀ d, y = fixacc d F R x W.
```

Consistency

Does it need decidable equality on T?

Conclusion

Using (non) terminating computations inside proofs

1. Prove "$\forall k,\ F^n\, k\, x = y$" by computation.
2. Compute then assume an accessibility witness.

Conclusion

Using (non) terminating computations inside proofs

1. Prove "$\forall k,\ F^n\, k\, x = y$" by computation.
2. Compute then assume an accessibility witness.

Questions?

- Can we get rid of the hypothesis "$\exists f,\ \forall x,\ P\, x\, (f\, x)$" in the generic lemma for the iterative version?
- Is the axiomatic version more useful in practice?
- It currently requires an ad-hoc proof for each F. Can we find a generic lemma as with the iterative version?