A Saucerful of Proofs in Coq

Olivier Danvy
Department of Computer Science
Aarhus University
danvy@cs.au.dk

Annapolis, Maryland 8 November 2012

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16, 25,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16, 25, 36,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16, 25, 36, 49,$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16, 25, 36, 49, 64, ...$$

Summing the first odd numbers

The stream of odd natural numbers:

$$1, 3, 5, 7, 9, 11, 13, 15, ...$$

The corresponding stream of partial sums:

$$1, 4, 9, 16, 25, 36, 49, 64, ...$$

i.e.,

$$1^2, 2^2, 3^2, 4^2, 5^2, 6^2, 7^2, 8^2, ...$$

Constructively

- start from the stream of natural numbers
- strike out every 2nd element
- compute the successive partial sums

Result: the stream of squares .

Scaling up

- start from the stream of natural numbers
- strike out every 3rd element
- compute the successive partial sums
- strike out every 2nd element
- compute the successive partial sums

Result: the stream of ...

Scaling up

- start from the stream of natural numbers
- strike out every 3rd element
- compute the successive partial sums
- strike out every 2nd element
- compute the successive partial sums

Result: the stream of cubes .

Scaling up: Moessner's theorem

- start from the stream of natural numbers
- strike out every nth element & sum
- strike out every $(n-1)$th element & sum
- ...
- strike out every 3rd element & sum
- strike out every 2nd element & sum

Result: the stream of powers of n .

Background

- Moessner (1951)

  The property, no proofs.

- Perron (1951), Paasche (1952), Salié (1952)

  Complicated inductive proofs.

- Hinze (IFL 2008)

  A calculational proof.

- Rutten & Niqui (HOSC 2012)

  A co-inductive proof.

This work (in progress)

- a formalization in Coq
- but first, learning Coq

Learning Coq in principle

- web resources
- book
- seasonal schools

Learning Coq in practice

- practice, practice, practice
- need a TA (or ideally, a coach)
- forces you to think things through
- can be frustrating at times

Asking an expert

```
Require Import Omega3.
```

Asking an expert

```
Require Import Omega3.

(* undocumented, but perfect here,
   thanks to the fatty acids: *)
do_the_right_thing.
```

Learning Coq in practice

- practice, practice, practice
- need a TA (or ideally, a coach)
- forces you to think things through
- can be frustrating at times
- wonderfully rewarding, overall

Teaching Coq

- introduction to functional programming (Q3 2011-2012, Q1 2012-2013)
- more advanced functional programming (Q4 2011-2012)

a marvelous experience

Term projects in Q3

- a standard batch (interpreters, compilers, decompilers, VMs, CPS, power series, searching in binary trees, Boolean negational normalization, FSA, etc.)

- a cherry on top of the pie: formalizing a theorem and a proof from another course(!)

Term projects in Q4

- functional & relational programming
- the Ackermann-Peter function
- Boolean normalization and equisatisfiability
- abstract interpretation (strided intervals)
- group theory and pronic numbers
- B-trees
- graph theory
- reduction from circuit to SAT
- vector spaces & Cauchy-Schwarz inequality

Plan

- Moessner's theorem at degree 3
- Moessner's theorem at degree 4
- Which starting indices in the master lemma?
- Introductory teaching with Coq
- Conclusion and perspectives

We are given

- `stream_of_ones`
- `stream_of_positive_powers_of_3`
- `stream_of_positive_powers_of_4`
- `skip_2`, `skip_3`, `skip_4`, ...
- `sums` & `sums_aux`, which uses an accumulator
- `stream_bisimilar`

Moessner's theorem at degree 3

- the statement
- the proof
- the master lemma

```
Theorem Moessner_3 :
  stream_bisimilar
    stream_of_positive_powers_of_3
    (sums
      (skip_2
        (sums
          (skip_3
            (sums
              stream_of_ones))))).
```

```
Proof.
  unfold stream_of_positive_powers_of_3.
  unfold sums.
  apply (Moessner_3_aux 0).
Qed.
```

```
Lemma Moessner_3_aux :
  forall (n : nat),
    stream_bisimilar
      (make_stream_of_nats (S n)
                           (fun i => i * i * i)
                           S)
      (sums_aux ???
        (skip_2
          (sums_aux ???
            (skip_3
              (sums_aux ???
                stream_of_ones))))).
```

Moessner's theorem at degree 4

- the statement
- the proof
- the master lemma

```
Theorem Moessner_4 :
  stream_bisimilar
    stream_of_positive_powers_of_4
    (sums
      (skip_2
        (sums
          (skip_3
            (sums
              (skip_4
                (sums
                  stream_of_ones))))))).
```

```
Proof.
  unfold stream_of_positive_powers_of_4.
  unfold sums.
  apply (Moessner_4_aux 0).
Qed.
```

```
Lemma Moessner_4_aux :
  forall (n : nat),
    stream_bisimilar
      (make_stream_of_nats (S n)
                           (fun i => i * i * i * i)
                           S)
      (sums_aux ???
         (skip_2
          (sums_aux ???
            (skip_3
              (sums_aux ???
                (skip_4
                  (sums_aux ???
                    stream_of_ones))))))).
```

So, which starting indices?

Newton's binomial expansion

Reminder:

$$
\begin{aligned}
(n+1)^2 &= n^2 + 2 \cdot n + 1 \\
(n+1)^3 &= n^3 + 3 \cdot n^2 + 3 \cdot n + 1 \\
(n+1)^4 &= n^4 + 4 \cdot n^3 + 6 \cdot n^2 + 4 \cdot n + 1 \\
&\ldots
\end{aligned}
$$

```
Lemma Moessner_2_aux :
  forall (n : nat),
    stream_bisimilar
      (make_stream_of_nats (S n)
                           (fun i => i * i)
                           S)
      (sums_aux  (n * n)
        (skip_2
          (sums_aux  (2 * n)
            stream_of_ ones  ))).
```

```
Lemma Moessner_3_aux :
  forall (n : nat),
    stream_bisimilar
      (make_stream_of_nats (S n)
                           (fun i => i * i * i)
                           S)
      (sums_aux (n * n * n)
        (skip_2
          (sums_aux (3 * n * n)
            (skip_3
              (sums_aux (3 * n)
                stream_of_ones ))))).
```

```
Lemma Moessner_4_aux :
  forall (n : nat),
    stream_bisimilar
      (make_stream_of_nats (S n)
                           (fun i => i * i * i * i)
                           S)
      (sums_aux (n * n * n * n)
         (skip_2
          (sums_aux (4 * n * n * n)
            (skip_3
              (sums_aux (6 * n * n)
                (skip_4
                  (sums_aux (4 * n)
                    stream_of_ones )))))))).
```

Assessment

- The monomials of the binomial expansion.
- Essential algebraic support from Coq

  to find the starting indices.
- A uniform structure for the proofs.
- A code generator in OCaml (Moe Masuko).
- A tactic in Ltac (Christian Clausen).
- And now what?

Consulting an expert – Danko Ilik

If your Ltac tactic does not use general recursion,

you can tease out the corresponding lambda-term.

This lambda-term is your proof.

Consulting an expert – Danko Ilik

If your Ltac tactic does not use general recursion,

you can tease out the corresponding lambda-term.

This lambda-term is your proof.

(The mathematical possibilities!)

My introductory lectures on Coq

The “what” is classical:

- functional programming, and proving

The “how” is classical too:

- from the known towards the unknown
- mathematical anxiety

From the known

Or more precisely, from what should be known:

- propositional logic
- proofs (e.g., Modus Ponens)

Strengthening the conclusion

```
Proposition modus_ponens_v1 :
  forall P Q : Prop,
    P /\ (P -> Q) -> Q.
Proof.
  intros P Q [H_P H_P_implies_Q].
  apply H_P_implies_Q.
  assumption.
Qed.
```

Weakening an hypothesis

```
Proposition modus_ponens_v2 :
  forall P Q : Prop,
    P /\ (P -> Q) -> Q.
Proof.
  intros P Q [H_P H_P_implies_Q].
  assert (H := H_P).
  apply H_P_implies_Q in H.
  assumption.
Qed.
```

Generalizing the goal

```
Proposition modus_ponens_v3 :
  forall P Q : Prop,
    P /\ (P -> Q) -> Q.
Proof.
  intros P Q [H_P H_P_implies_Q].
  revert H_P.
  assumption.
Qed.
```

From the known

Or more precisely, from what should be known:

- propositional logic
- proofs (e.g., Modus Ponens)
- inductive definition of data
- recursive definition of programs

Conclusion and perspectives

Proof assistants are changing the world:

- the 4-color theorem
- the FeitThompson theorem
- DemTech

This was then: MFPS 1991

This was then: MFPS 1991

"Look guys.

We don't say anything about your proofs,

so don't say anything about our programs,

OK?"

This is now: MAP 2012

Moi: A lightweight question: how does it feel, ...

This is now: MAP 2012

Moi: A lightweight question: how does it feel, as a mathematician, to show your proofs to the world in complete detail?

This is now: MAP 2012

Moi: A lightweight question: how does it feel, as a mathematician, to show your proofs to the world in complete detail?

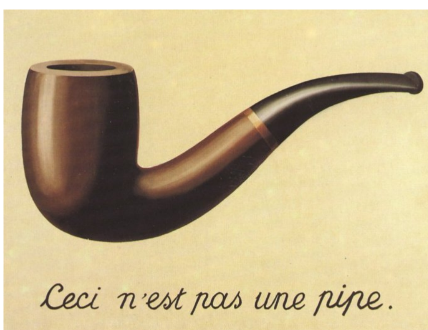
V. Voevodsky (smiling): Liberating.

This is now: MAP 2012

Moi: A lightweight question: how does it feel, as a mathematician, to show your proofs to the world in complete detail?

V. Voevodsky (smiling): Liberating.

Y. Bertot (intervening): But you only show a representation of your proofs.



This is now: MAP 2012

Moi: A lightweight question: how does it feel, as a mathematician, to show your proofs to the world in complete detail?

V. Voevodsky (smiling): Liberating.

Y. Bertot (intervening): But you only show a representation of your proofs.

This is now: MAP 2012

Moi: A lightweight question: how does it feel,
as a mathematician, to show your proofs
to the world in complete detail?

V. Voevodsky (smiling): Liberating.

Y. Bertot (intervening): But you only show
a representation of your proofs.
Their spirit might be lost in the process.

This is now: MAP 2012

Moi: A lightweight question: how does it feel,
as a mathematician, to show your proofs
to the world in complete detail?

V. Voevodsky (smiling): Liberating.

Y. Bertot (intervening): But you only show
a representation of your proofs.
Their spirit might be lost in the process.

V. Voevodsky (definitely): <u>I doubt it</u>.

This is now: MAP 2012

Moi: A lightweight question: how does it feel,
as a mathematician, to show your proofs
to the world in complete detail?

V. Voevodsky (smiling): Liberating.

Y. Bertot (intervening): But you only show
a representation of your proofs.
Their spirit might be lost in the process.

V. Voevodsky (definitely): <u>I doubt it</u>.

Thank you.