Home
PUBLIC
Questions
Tags
Users
Companies
COLLECTIVES
Explore Collectives
TEAMS
Stack Overflow for Teams – Start collaborating and sharing organizational knowledge.
Create a free Team
Why Teams?

How to avoid "Cannot guess decreasing argument of fix." in Coq

Ask Question

Asked yesterday Modified today Viewed 39 times

```
(* Define a function sum_digits such that its input is an int,  the base of which
   output is the sum of all its digits *)

Definition sum_digits (digits : nat) (base : nat) : nat :=
  let fix aux (sum : nat) (x_value : nat) : nat :=
    if leb x_value 0 then
      sum
    else
      aux (add sum (modulo x_value base)) (div x_value base) in
  aux 0 digits.
```

Error: Cannot guess decreasing argument of fix.

I Defined a function according to the description above, but it seems that Coq cannot deal with this kinds of recursion. How could I fix this problem?

recursion coq coqide

Share Edit Delete Flag

asked yesterday
Ernie2C_A
1
New contributor

Add a comment

3 Answers

Sorted by: Highest score (default)

`fix` can be used with structural recursion. But `(div x_value base)` is not a subterm of `x_value`. You may use `Equations` to define your function through a well-foundedness argument.

```
Require Import Arith.
Import Nat.
From Equations Require Import Equations.

Section Base.
Variable base: nat.
Hypothesis Hbase : 1 < base.

Equations sum_aux (digits: nat) (sum: nat): nat  by wf digits lt :=
 sum_aux 0 sum := sum;
 sum_aux d sum := sum_aux (div d base)
                          (add sum (modulo d base)).
Next Obligation.
 apply div_lt; auto with arith.
Defined.

Definition sum_digits (digits : nat) :=
  sum_aux digits 0.

End Base.

Compute sum_digits 10 _ 231.
```

Share Edit Follow edited 19 hours ago answered 20 hours ago
Pierre Castéran
521 • 2 • 5

Add a comment

You can use the `Program` library. It can sometimes be a little tricky to use, but in this case it works fine. You enter your program, and then you get a bunch of obligations to solve. `Next Obligation.` gives you the next obligation to solve.

Here we say `{measure digits}` which means that `digits` will be strictly decreasing. This will guarantee that the execution terminates.

```
Require Import Program Nat.

Program Fixpoint sum_digits (digits : nat) (base : nat) (sum : nat) (measure digi
if dec (eqb digits 0) then sum
else sum_digits (digits/base) base (sum + modulo digits base).

Next Obligation.
```

Note that I write `dec (eqb digits 0)` so that we don't forget that `digits != 0` in the second branch.

When we do `Next Obligation.` we get the goal

```
digits / base < digits
```

Let's search for a proof of this.

```
Search (?a / ?b < ?a).

==>  PeanoNat.Nat.div_lt: forall a b : nat, 0 < a -> 1 < b -> a / b < a
```

Ouch. This is something that we can't prove unless we also know that `1 < base`. Let's add an argument `Hbase: 1<base`.

Change that, and apply `div_lt`. Finally use `(digits =? 0) = false` to prove that `0 < digits`. Search for something that can help us.

```
Search ((?a =? ?b) = false).

==> PeanoNat.Nat.eqb_neq: forall x y : nat, (x =? y) = false <-> x <> y
```

Lets apply that. And we're essentially done.

Here is the complete session.

```
Require Import Program Nat.

Program Fixpoint sum_digits (digits : nat) (base :nat) (Hbase: 1 < base) (sum : n
if dec (eqb digits 0) then sum else sum_digits (digits/base) base Hbase (sum + mo

Next Obligation.
  apply PeanoNat.Nat.div_lt.
  apply PeanoNat.Nat.eqb_neq in e.
  destruct digits. exfalso; congruence. apply PeanoNat.Nat.lt_0_succ.
  apply Hbase.
Defined.
```

Try it with

```
Compute sum_digits 758 2 _ 0.

= 7
  : nat
```

Share Edit Follow edited 15 hours ago answered 15 hours ago
larsr
5,381 • 18 • 38

Add a comment

I assume you work in a context where `leb` is a function with type `nat -> nat -> bool`. Guessing from its name, you probably inherit it from importing the `Arith` module. I also assume you have a function `div` of type `nat -> nat -> nat`.

The typing algorithm does not know enough about the `leb` function. As far as typing is concerned, this function might just return `false` all the time. So, in the `else` branch, it may be that `x_value` is 0, and if this is the case, `(div x_value base)` is not smaller than `x_value`.

Now, even if the typing algorithm was able to understand what happens in the division algorithm, guaranteeing that an argument decreases would still not be possible, because there is a case where division does not return a smaller value: when base is 1.

So the code you have in mind is clear from the perspective of a human reader, but it is based on assumptions that need to be clarified at the time of writing the code.

How to fix the problem

- A first solution is to use a more advanced tool to define your recursive function, one where you will be able to use explicit proofs to explain why arguments are decreasing. Here is my attempt:

```
Require Import Arith.
From Equations Require Import Equations.

Import Nat.

Definition inspect {A} (a : A) : {b | a = b} :=
  exist _ a refl_equal.

Notation "x 'eqn:' p" := (exist _ x p) (only parsing, at level 20).


Equations sum_digits_aux (sum : nat) (x_value : nat) (base: nat)
  (h : 1 < base) :
  nat by wf x_value lt :=
sum_digits_aux sum x_value base base_gt_1 with inspect (leb x_value 0) := {
| true eqn:x_value_is_0 => sum
| false eqn:x_value_n_0 =>
  sum_digits_aux (add sum (modulo x_value base)) (div x_value base) base
    base_gt_1
  }.
Next Obligation.
Proof.
apply div_lt.
  rewrite leb_iff_conv in x_value_n_0.
  easy.
easy.
Qed.

Definition sum_digits (digits : nat) (base : nat) : nat :=
  match le_lt_dec base 1 with
  | left base_le_1 => 0
  | right base_gt_1 =>
    sum_digits_aux 0 digits base base_gt_1
  end.
```

Note that the definition of `sum_digits_aux` basically rephrases the algorithm you gave in your definition for the `aux` recursive function. There is a trick the `inspect` pattern to make sure we have an hypotheses that tells in which branch of the if-then-else construct we are. Hypothesis `x_value_n_0` is important for the proofs that come later.

The `by wf x_value Peano.lt` fragment specifies that the "decrease" notion will be the one of the strict order for natural numbers. This order is known to be well-founded.

Note also that my `sum_digits_aux` takes one more argument when compared to yours: I need a proof that `base` is strictly larger than 1, otherwise, I cannot guarantee the decrease.

When the `Equations` command is processed, the system understands the algorithm is still cannot prove that the `x_value` argument decreases by itself, so it give that as an obligation to be proved. I show here how to prove it, using theorems that are present in Coq libraries.

- A second solution is to use the basic knowledge handled by the guard checker: rely only on pattern matching and use a fuel argument.

```
Definition sum_digits' (digits : nat) (base : nat) : nat :=
  let fix aux (sum x_value base fuel : nat) : nat :=
    match fuel with
    | 0 => sum
    | S p =>
      if leb x_value 0 then
        sum
      else
        aux (add sum (modulo x_value base)) (div x_value base) base p
    end in
  aux 0 digits base digits.
```

I don't like this solution, because the fuel argument is artificial and it is not clear what happens if the base is 1 or the fuel is strictly smaller than `x_value`, but these cases are not *meant to happen*. It can be a source of bugs.

Whether you call this function with a base that is larger than 1 or not is up to you. If you call this function with a base that is larger than 1, then `fuel` is guaranteed to decrease by only 1 at each recursive call, while `x_value` is divided by `base` at each recursive call, so `fuel` should remain larger than `x_value`. Ultimately, you should be able to prove in Coq that the two functions defined in this solution `sum_digits` and `sum_digits'` behave the same whenever `1 < base`.

[EDIT]: in a previous version of `sum_digits'`, the value returned when `fuel` is 0 was 0, this is wrong and leads to `sum_digits' 1 2` being wrongly computed.

Share Edit Follow edited 13 hours ago answered 19 hours ago
Yves
3,608 • 12 • 12

Add a comment

Answer Your Question

The Overflow Blog
Let's talk about our favorite terminal tools (Ep. 521)
Best practices to increase the speed for Next.js apps
Featured on Meta
Help us identify new roles for community members
Navigation and UI research starting soon
2022 Community Moderator Election Results - now with two more mods!
Proposing a Community-Specific Closure Reason for non-English content
Temporary policy: ChatGPT is banned
I'm standing down as a moderator

Related

2 How to indicate decreasing in size of two Coq inductive types
4 Well founded recursion in Coq
5 Cannot guess decreasing argument of fix for nested match in Coq
2 Error: Cannot guess decreasing argument of fix. Coq
1 Building up tree and decreasing argument of fix
6 Defining recursive function over product type
1 Implement Boruvka's algorithm in Coq
0 Coq: Cannot guess decreasing argument of fix
1 Problems adding new notation in Coq
1 COQ returns cannot guess decreasing argument of fix. How to fix my function?

Hot Network Questions

Draw 2 stacking was wrong or official account played a prank?
What is the QGD set-up with queen-side fianchetto and c4 called?
8-letter word that uses all letter keys on a telephone
Did Kim Clement predict the Russo-Ukrainian War?
Is self-defense allowed when there are objectively reasonable grounds but it is actually done subjectively for improper reasons?
Plus minus reversal in the post length decoration
Is there enough matter orbiting the sun to make a second sun?
Dimension of a 3.5-inch floppy disk
How to enter a multi-line command in bash on command line
How do you fit so many instructions on a 8-bit processor?
What is the meaning of the word "fermison"?
Did Lincoln say, "When we buy steel made in America, we get the steel and the money too"?
Do aromatic canonical structures make the overall compound aromatic?
If Mars and Venus were habitable, by when would they have been colonized?
Does Elon Musk have any grounds for legal action against people who track his plane?
How Do I Change My Guitar Tuning?
How to find the nearest palette color when dithering in RGB
Of nouns that can be one of two genders, are there any that can't be masculine?
Should one state the sharpest version of a Lemma even if only a weaker version is needed?
Sum every second digit in a number
How do zombies survive for more than just a few months?
What is the minimum light intensity that a human eye can detect?
Replace all vowels with repeated "aeiou"
Can 777-characters long passphrase be considered too short?
Question feed