

Fun With Functions

Tobias Nipkow

August 7, 2022

Abstract

This is a collection of cute puzzles of the form “Show that if a function satisfies the following constraints, it must be ...” Please add further examples to this collection!

Apart from the one about factorial, they all come from the delightful booklet by Terence Tao [1] but go back to Math Olympiads and similar events.

Please add further examples of this kind, either directly or by sending them to me. Let us make this a growing body of *fun!*

theory *FunWithFunctions* **imports** *Complex-Main* **begin**

See [1]. Was first brought to our attention by Herbert Ehler who provided a similar proof.

theorem *identity1*: **fixes** $f :: nat \Rightarrow nat$

assumes $fff: \bigwedge n. f(f(n)) < f(Suc(n))$

shows $f(n) = n$

proof –

{ **fix** $m\ n$ **have** *key*: $n \leq m \implies n \leq f(m)$

proof(*induct n arbitrary: m*)

case 0 **show** *?case* **by** *simp*

next

case (*Suc n*)

hence $m \neq 0$ **by** *simp*

then obtain k **where** [*simp*]: $m = Suc\ k$ **by** (*metis not0-implies-Suc*)

have $n \leq f(k)$ **using** *Suc* **by** *simp*

hence $n \leq f(f(k))$ **using** *Suc* **by** *simp*

also have $\dots < f(m)$ **using** *fff* **by** *simp*

finally show *?case* **by** *simp*

qed }

hence $\bigwedge n. n \leq f(n)$ **by** *simp*

hence $\bigwedge n. f(n) < f(Suc\ n)$ **by**(*metis fff order-le-less-trans*)

hence $f(n) < n+1$ **by** (*metis fff lift-Suc-mono-less-iff[of f] Suc-eq-plus1*)

with $\langle n \leq f(n) \rangle$ **show** $f\ n = n$ **by** *arith*

qed

See [1]. Possible extension: Should also hold if the range of f is the reals!

```

lemma identity2: fixes  $f :: nat \Rightarrow nat$ 
assumes  $f(k) = k$  and  $k \geq 2$ 
and  $f$ -times:  $\bigwedge m n. f(m*n) = f(m)*f(n)$ 
and  $f$ -mono:  $\bigwedge m n. m < n \implies f m < f n$ 
shows  $f(n) = n$ 
proof -
  have 0:  $f(0) = 0$ 
    by (metis  $f$ -mono  $f$ -times mult-1-right mult-is-0 nat-less-le nat-mult-eq-cancel-disj
not-less-eq)
  have 1:  $f(1) = 1$ 
    by (metis  $f$ -mono  $f$ -times gr-implies-not0 mult-eq-self-implies-10 nat-mult-1-right
zero-less-one)
  have 2:  $f 2 = 2$ 
    proof -
      have  $2 + (k - 2) = k$  using  $\langle k \geq 2 \rangle$  by arith
      hence  $f(2) \leq 2$ 
        using mono-nat-linear-lb[of  $f 2 k - 2$ , OF  $f$ -mono]  $\langle f k = k \rangle$ 
        by simp
      thus  $f 2 = 2$  using 1  $f$ -mono[of 1 2] by arith
    qed
  show ?thesis
proof (induct rule:less-induct)
  case (less  $i$ )
  show ?case
  proof cases
    assume  $i \leq 1$  thus ?case using 0 1 by (auto simp add:le-Suc-eq)
  next
    assume  $\sim i \leq 1$ 
    show ?case
    proof cases
      assume  $i \bmod 2 = 0$ 
      hence  $\exists k. i = 2*k$  by arith
      then obtain  $k$  where  $i = 2*k$  ..
      hence  $0 < k$  and  $k < i$  using  $\langle \sim i \leq 1 \rangle$  by arith+
      hence  $f(k) = k$  using less(1) by blast
      thus  $f(i) = i$  using  $\langle i = 2*k \rangle$  by (simp add:f-times 2)
    next
      assume  $i \bmod 2 \neq 0$ 
      hence  $\exists k. i = 2*k + 1$  by arith
      then obtain  $k$  where  $i = 2*k + 1$  ..
      hence  $0 < k$  and  $k + 1 < i$  using  $\langle \sim i \leq 1 \rangle$  by arith+
      have  $2*k < f(2*k + 1)$ 
      proof -
        have  $2*k = 2*f(k)$  using less(1)  $\langle i = 2*k + 1 \rangle$  by simp
        also have  $\dots = f(2*k)$  by (simp add:f-times 2)
        also have  $\dots < f(2*k + 1)$  using  $f$ -mono[of  $2*k$   $2*k + 1$ ] by simp
        finally show ?thesis .
      qed
    qed
  qed

```

```

moreover
have  $f(2*k+1) < 2*(k+1)$ 
proof -
  have  $f(2*k+1) < f(2*k+2)$  using  $f\text{-mono}[of\ 2*k+1\ 2*k+2]$  by  $simp$ 
  also have  $\dots = f(2*(k+1))$  by  $simp$ 
  also have  $\dots = 2*f(k+1)$  by( $simp\ only:f\text{-times}\ 2$ )
  also have  $f(k+1) = k+1$  using  $less(1)\ \langle i=2*k+1 \rangle\ \langle \sim i \leq 1 \rangle$  by  $simp$ 
  finally show  $?thesis$  .
qed
ultimately show  $f(i) = i$  using  $\langle i = 2*k+1 \rangle$  by  $arith$ 
qed
qed
qed
qed

```

One more from Tao's booklet. If f is also assumed to be continuous, $f x = x + 1$ holds for all reals, not only rationals. Extend the proof!

```

theorem plus1:
fixes  $f :: real \Rightarrow real$ 
assumes  $0: f\ 0 = 1$  and  $f\text{-add}: \bigwedge x\ y. f(x+y+1) = f\ x + f\ y$ 

assumes  $r : \mathbb{Q}$  shows  $f(r) = r + 1$ 
proof -
  { fix  $i$  have  $f(of\text{-int}\ i) = of\text{-int}\ i + 1$ 
    proof ( $induct\ i\ rule: int\text{-induct}\ [where\ k=0]$ )
      case  $base$  show  $?case$  using  $0$  by  $simp$ 
    next
      case ( $step1\ i$ )
      have  $f(of\text{-int}\ (i+1)) = f(of\text{-int}\ i + 0 + 1)$  by  $simp$ 
      also have  $\dots = f(of\text{-int}\ i) + f\ 0$  by( $rule\ f\text{-add}$ )
      also have  $\dots = of\text{-int}\ (i+1) + 1$  using  $step1\ 0$  by  $simp$ 
      finally show  $?case$  .
    next
      case ( $step2\ i$ )
      have  $f(of\text{-int}\ i) = f(of\text{-int}\ (i - 1) + 0 + 1)$  by  $simp$ 
      also have  $\dots = f(of\text{-int}\ (i - 1)) + f\ 0$  by( $rule\ f\text{-add}$ )
      also have  $\dots = f(of\text{-int}\ (i - 1)) + 1$  using  $0$  by  $simp$ 
      finally show  $?case$  using  $step2$  by  $simp$ 
    qed }
  note  $f\text{-int} = this$ 
  { fix  $n\ r$  have  $f(of\text{-int}\ (Suc\ n)*r + of\text{-int}\ n) = of\text{-int}\ (Suc\ n) * f\ r$ 
    proof( $induct\ n$ )
      case  $0$  show  $?case$  by  $simp$ 
    next
      case ( $Suc\ n$ )
      have  $of\text{-int}\ (Suc(Suc\ n))*r + of\text{-int}\ (Suc\ n) =$ 
         $r + (of\text{-int}\ (Suc\ n)*r + of\text{-int}\ n) + 1$  (is  $?a = ?b$ )
      by( $simp\ add: field\text{-simps}$ )
      hence  $f\ ?a = f\ ?b$ 
    }

```

```

    by presburger
    also have ... = f r + f(of-int (Suc n)*r + of-int n) by(rule f-add)
    also have ... = f r + of-int (Suc n) * f r by(simp only:Suc)
    finally show ?case by(simp add: field-simps)
  qed }
note 1 = this
{ fix n::nat and r assume n≠0
  have f(of-int (n)*r + of-int (n - 1)) = of-int (n) * f r
  proof(cases n)
    case 0 thus ?thesis using ⟨n≠0⟩ by simp
  next
    case Suc thus ?thesis using ⟨n≠0⟩ using 1 by auto
  qed }
note f-mult = this
from ⟨r:ℚ⟩ obtain i::int and n::nat where r: r = of-int i/of-int n and n≠0
  by(fastforce simp:Rats-eq-int-div-nat)
have of-int (n) * f(of-int i / of-int n) = f(of-int i + of-int (n - 1))
  using ⟨n≠0⟩
  by (metis (no-types, opaque-lifting) f-mult mult.commute nonzero-divide-eq-eq
of-int-of-nat-eq of-nat-0-eq-iff)
  also have ... = f(of-int (i + int n - 1)) using ⟨n≠0⟩[simplified]
  by (metis One-nat-def Suc-leI of-nat-1 add-diff-eq of-int-add of-nat-diff)
  also have ... = of-int (i + int n - 1) + 1 by(rule f-int)
  also have ... = of-int i + of-int n by arith
  finally show ?thesis using ⟨n≠0⟩ unfolding r by (simp add:field-simps)
qed

```

The only total model of a naive recursion equation of factorial on integers is 0 for all negative arguments. Probably folklore.

```

theorem ifac-neg0: fixes ifac :: int ⇒ int
assumes ifac-rec:  $\bigwedge i. \text{ifac } i = (\text{if } i=0 \text{ then } 1 \text{ else } i*\text{ifac}(i - 1))$ 
shows  $i < 0 \implies \text{ifac } i = 0$ 
proof(rule ccontr)
  assume 0:  $i < 0 \text{ ifac } i \neq 0$ 
  { fix j assume  $j \leq i$ 
    have ifac j ≠ 0
      apply(rule int-le-induct[OF ⟨j≤i⟩])
      apply(rule ⟨ifac i ≠ 0⟩)
      apply (metis ⟨i<0⟩ ifac-rec linorder-not-le mult-eq-0-iff)
      done
  } note below0 = this
{ fix j assume  $j < i$ 
  have  $1 < -j$  using ⟨j<i⟩ ⟨i<0⟩ by arith
  have ifac(j - 1) ≠ 0 using ⟨j<i⟩ by(simp add: below0)
  then have  $|\text{ifac } (j - 1)| < (-j) * |\text{ifac } (j - 1)|$  using ⟨j<i⟩
    mult-le-less-imp-less[OF order-refl[of abs(ifac(j - 1))] ⟨1 < -j⟩]
    by(simp add:mult.commute)
  hence  $\text{abs}(\text{ifac}(j - 1)) < \text{abs}(\text{ifac } j)$ 
    using ⟨1 < -j⟩ by(simp add: ifac-rec[of j] abs-mult)

```

```

} note not-wf = this
let ?f = %j. nat(abs(ifact(i - int(j+1))))
obtain k where  $\neg ?f (Suc\ k) < ?f\ k$ 
  using wf-no-infinite-down-chainE[OF wf-less, of ?f] by blast
moreover have  $i - int\ (k + 1) - 1 = i - int\ (Suc\ k + 1)$  by arith
ultimately show False using not-wf[of i - int(k+1)]
  by (simp only:) arith
qed

end

```

References

- [1] Terence Tao. *Solving Mathematical Problems*. Oxford University Press, 2006.