OM 1969 — Stage 1

Jakub Kądziołka

October 13, 2020

Contents

1 Warmup problems (Series I)

Long ago, the Polish Math Olympiad published, apart from 12 problems to be solved and mailed over 3 months, a set of 12 warmup problems, which were similar in spirit, but easier.

theory *WarmupI*
  imports
    Complex-Main
    Future-Library.Future-Library
    HOL−Library.Sum-of-Squares
    HOL−Library.Quadratic-Discriminant
    HOL−Number-Theory.Cong
    HOL−Analysis.Analysis
begin

1.1 Warmup 1

Solve the equation $3^x = 4y + 5$ in the integers.

We begin with the following lemma:

lemma *even-power-3*: $[3\,\hat{}\,k = 1::int]$ $(mod\ 4) \longleftrightarrow even\ k$
proof −

```
  have [3^k = (−1::int)^k] (mod 4)
    by (intro cong-pow) (auto simp: cong-def)
  thus ?thesis
    by (auto simp: cong-def minus-one-power-iff)
qed
```

Here is an alternative proof — hopefully it will be instructive in doing calculations mod n.

```
lemma [3^k = 1::int] (mod 4) ⟷ even k
proof (cases even k)
  case True
  then obtain l where 2*l = k by auto
  then have [3^k = (3^2)^l] (mod 4) (is cong - ... -)
    by (auto simp add: power-mult)
  also have [... = (1::int)^l] (mod 4) (is cong - ... -)
    by (intro cong-pow) (simp add: cong-def)
  also have [... = 1] (mod 4) by auto
  finally have [3^k = 1::int] (mod 4).
  thus ?thesis using ‹even k› by blast
next
  case False
  then obtain l where 2*l + 1 = k
    using oddE by blast
  then have [3^k = 3^(2*l+1)] (mod 4) (is cong - ... -) by auto
  also have [... = (3^2)^l * 3] (mod 4) (is cong - ... -)
    by (metis power-mult power-add power-one-right cong-def)
  also have [... = (1::int)^l * 3] (mod 4) (is cong - ... -)
    by (intro cong-mult cong-pow) (auto simp add: cong-def)
  also have [... = 3] (mod 4) by auto
  finally have [3^k ≠ 1::int] (mod 4) by (auto simp add: cong-def)
  then show ?thesis using ‹odd k› by blast
qed
```

This allows us to prove the theorem, provided we assume x is a natural number.

```
theorem warmup1-natx:
  fixes x :: nat and y :: int
  shows 3^x = 4*y + 5 ⟷ even x ∧ y = (3^x − 5) div 4
proof −
  have even x ∧ y = (3^x − 5) div 4 if 3^x = 4*y + 5
  proof −
    from that have [3^x = 4*y + 5] (mod 4) by auto
    also have [4*y + 5 = 5] (mod 4)
      by (metis cong-mult-self-left cong-add-rcancel-0)
    also have [5 = 1::int] (mod 4) by (auto simp add: cong-def)
    finally have [(3::int)^x = 1] (mod 4).
    hence even x using even-power-3 by auto
    thus ?thesis using that by auto
  qed
```

moreover have *3 ^ x = 4 * y + 5* **if** *even x ∧ y = (3^x − 5) div 4*
proof −
from *that* **have** *even x* **and** *y-form: y = (3^x − 5) div 4* **by** *auto*
then have *[3^x = 1::int] (mod 4)* **using** *even-power-3* **by** *blast*
then have *((3::int)^x − 5) mod 4 = 0*
by *(simp add: cong-def mod-diff-cong)*
thus *?thesis* **using** *y-form* **by** *auto*
qed
ultimately show *?thesis* **by** *blast*
qed

To consider negative values of x, we'll need to venture into the reals:

lemma *powr-int-pos*:
fixes *x y :: int*
assumes *∗: 3 powr x = y*
shows *x ≥ 0*
proof *(rule ccontr)*
assume *neg-x: ¬ x ≥ 0*
then have *y-inv: y = inverse ((3::nat)^nat (−x))* (**is** *y = inverse (?n::nat)*)
using *powr-real-of-int* **and** *∗* **by** *auto*
hence *real ?n ∗ of-int y = 1* **by** *auto*
hence *?n ∗ y = 1* **using** *of-int-eq-iff* **by** *fastforce*
hence *?n = 1*
by *(metis nat-1-eq-mult-iff nat-int nat-numeral-as-int numeral-One of-nat-mult zmult-eq-1-iff)*
hence *nat (−x) = 0* **by** *auto*
thus *False* **using** *neg-x* **by** *auto*
qed

corollary *warmup1*:
fixes *x y :: int*
shows *3 powr x = 4∗y + 5 ⟷ x ≥ 0 ∧ even x ∧ y = (3^(nat x) − 5) div 4*
proof
assume *assm: 3 powr x = 4∗y + 5*
then have *x ≥ 0* **using** *powr-int-pos* **by** *fastforce*
hence *3 powr (nat x) = 4∗y + 5* **using** *assm* **by** *simp*
hence *(3::real)^(nat x) = 4∗y + 5* **using** *powr-realpow* **by** *auto*
hence *with-nat: 3^(nat x) = 4∗y + 5* **using** *of-int-eq-iff* **by** *fastforce*
hence *even (nat x) ∧ y = (3^(nat x) − 5) div 4* **using** *warmup1-natx* **by** *auto*
thus *x ≥ 0 ∧ even x ∧ y = (3^(nat x) − 5) div 4* **using** ‹*x ≥ 0*› **and** *even-nat-iff* **by** *auto*
next
assume *assm: x ≥ 0 ∧ even x ∧ y = (3^(nat x) − 5) div 4*
then have *3^(nat x) = 4∗y + 5* **using** *warmup1-natx* **and** *even-nat-iff* **by** *blast*
thus *3 powr x = 4∗y + 5* **using** *assm powr-real-of-int* **by** *fastforce*
qed

1.2 Warmup 2

Prove that, for all real a and b we have

$$(a + b)^4 \leq 8(a^4 + b^4).$$

This problem is simple enough for Isabelle to solve it automatically — with the Sum of Squares decision procedure.

```
theorem
  (a+b)^4 ≤ 8*(a^4 + b^4) for a b :: real
  by sos
```

Of course, we would rather elaborate. We will make use of the inequality known as *sum-squares-bound*:

$(2::'a) * x * y \leq x^2 + y^2$

```
theorem
  (a+b)^4 ≤ 8*(a^4 + b^4) for a b :: real
proof −
  have lemineq: 2*x^3*y ≤ x^4 + x^2*y^2 for x y :: real
    using sum-squares-bound [of x y]
      and mult-left-mono [where c=x^2]
    by (force simp add: numeral-eq-Suc algebra-simps)

  have (a+b)^4 = a^4 + 4*a^3*b + 6*a^2*b^2 + 4*a*b^3 + b^4 by algebra
 also have ... ≤ a^4 + 2*(a^4 + a^2*b^2) + 6*a^2*b^2 + 2*(b^4 + a^2*b^2)
+ b^4
    using lemineq [of a b]
      and lemineq [of b a]
    by (simp add: algebra-simps)
  also have ... = 3*a^4 + 3*b^4 + 10*a^2*b^2 by (simp add: algebra-simps)
  also have ... ≤ 8*(a^4 + b^4)
    using sum-squares-bound [of a^2 b^2]
    by simp
  finally show ?thesis.
qed
```

Another interesting proof is by Jensen's inequality. In Isabelle, it's known as the *convex-on* lemma:

```
convex S ⟹
convex-on S f =
(∀k u x.
    (∀i∈{1..k}. 0 ≤ u i ∧ x i ∈ S) ∧ sum u {1..k} = 1 ⟶
    f (∑i = 1..k. u i *R x i) ≤ (∑i = 1..k. u i * f (x i)))
```

Note that the sequences u and x are modeled as functions $nat \Rightarrow real$, thus instead of u_i we have $u\ i$.

Make sure not to confuse the *convex-on* lemma with the *convex-on* predicate, which is defined by *convex-on-def*:

```
convex-on s f =
(∀x∈s. ∀y∈s. ∀u≥0. ∀v≥0. u + v = 1 ⟶
                                    f (u *R x + v *R y) ≤ u * f x + v * f y)
```

The bulk of the work, of course, is in showing that our function, $x \mapsto x^4$, is convex.

```
theorem warmup2:
  (a+b)^4 ≤ 8*(a^4 + b^4) for a b :: real
proof −
  let ?f = λx. x^4
  have convex-on UNIV ?f
  proof (rule f''-ge0-imp-convex)
    show convex UNIV by auto
    let ?f' = λx. 4*x^3
    show (?f has-real-derivative ?f' x) (at x) for x :: real
      using DERIV-pow [where n=4] by fastforce
    let ?f'' = λx. 12*x^2
    show (?f' has-real-derivative ?f'' x) (at x) for x :: real
      using DERIV-pow [where n=3]
        and DERIV-cmult [where c=4]
      by fastforce
    show 0 ≤ ?f'' x for x :: real
      by auto
  qed
  hence (a/2 + b/2)^4 ≤ a^4/2 + b^4/2 (is ?lhs ≤ ?rhs)
    using convex-onD [where t=1/2] by fastforce
  also have ?lhs = ((a + b)/2)^4 by algebra
   also have ... = (a+b)^4/16 using power-divide [of a+b 2, where n=4] by
fastforce
  finally show ?thesis by auto
qed
```

1.3 Warmup 3

This one is a straight-forward equation:

```
theorem warmup3:
  |x−1|*|x+2|*|x−3|*|x+4| = |x+1|*|x−2|*|x+3|*|x−4|
    ⟷ x ∈ {0, sqrt 7, −sqrt 7,
              sqrt ((13 + sqrt 73) / 2),
             −sqrt ((13 + sqrt 73) / 2),
              sqrt ((13 − sqrt 73) / 2),
             −sqrt ((13 − sqrt 73) / 2)}
  (is ?eqn ⟷ ?sols)
proof −
  have ?eqn ⟷ |(x−1)*(x+2)*(x−3)*(x+4)| = |(x+1)*(x−2)*(x+3)*(x−4)|
(is - ⟷ |?lhs| = |?rhs|)
```

```
    by (simp add: abs-mult)
   also have ... ⟷ ?lhs − ?rhs = 0 ∨ ?lhs + ?rhs = 0 by (auto simp add: abs-eq-iff)
  also have ... ⟷ x∗(x^2 − 7) = 0 ∨ x^4 − 13∗x^2 + 24 = 0 by algebra
  also have x∗(x^2 − 7) = 0 ⟷ x ∈ {0, sqrt 7, −sqrt 7} using plus-or-minus-sqrt by auto
  also have x^4 − 13∗x^2 + 24 = 0 ⟷ x² ∈ {(13 + sqrt 73) / 2, (13 − sqrt 73) / 2}
    using discriminant-nonneg [where x=x^2, of 1 −13 24]
    by (auto simp add: algebra-simps discrim-def)
  also have ... ⟷ x ∈ {sqrt ((13 + sqrt 73) / 2),
                      −sqrt ((13 + sqrt 73) / 2),
                       sqrt ((13 − sqrt 73) / 2),
                      −sqrt ((13 − sqrt 73) / 2)}
  proof −
    have 0 ≤ (13 − sqrt 73) / 2 by (auto simp add: real-le-lsqrt)
    hence x² = (13 − sqrt 73) / 2
          ⟷ x ∈ {sqrt ((13 − sqrt 73) / 2),
                −sqrt ((13 − sqrt 73) / 2)}
      using plus-or-minus-sqrt
      by blast
    moreover have x² = (13 + sqrt 73) / 2
      ⟷ x ∈ {sqrt ((13 + sqrt 73) / 2),
            −sqrt ((13 + sqrt 73) / 2)}
        by (smt insert-iff power2-minus power-divide real-sqrt-abs real-sqrt-divide real-sqrt-pow2 singletonD)
    ultimately show ?thesis by blast
  qed
  ultimately show ?thesis by blast
qed
```

1.4 Warmup 4

There is a set of n points on a plane with the property that, in each triplet of points, there's a pair with distance at most 1. Prove that the set can be covered with two circles of radius 1.

There's nothing special about the case of points on a plane, the theorem can be proved without additional difficulties for any metric space:

```
theorem warmup4-generic:
  fixes S :: 'a::metric-space set
  assumes finite S
  assumes property: ⋀T. T ⊆ S ∧ card T = 3 ⟹ ∃p∈T. ∃q∈T. p ≠ q ∧ dist p q ≤ 1
  obtains O₁ O₂ where S ⊆ cball O₁ 1 ∪ cball O₂ 1
proof
  let ?pairs = S × S
  let ?dist = λ(a, b). dist a b
  define widest-pair where widest-pair = arg-max-on ?dist ?pairs
```

```
let ?O₁ = (fst widest-pair)
let ?O₂ = (snd widest-pair)
show S ⊆ cball ?O₁ 1 ∪ cball ?O₂ 1
proof
  fix x
  assume x ∈ S

  from ‹finite S› and ‹x ∈ S›
  have finite ?pairs and ?pairs ≠ {} by auto
  hence OinS: widest-pair ∈ ?pairs
    unfolding widest-pair-def by (simp add: arg-max-if-finite)

  have ∀(P,Q)∈?pairs. dist ?O₁ ?O₂ ≥ dist P Q
    unfolding widest-pair-def
    using ‹finite ?pairs› and ‹?pairs ≠ {}›
    by (metis (mono-tags, lifting) arg-max-greatest prod.case-eq-if)
  hence greatest: dist P Q ≤ dist ?O₁ ?O₂ if P ∈ S and Q ∈ S for P Q
    using that by blast

  let ?T = {?O₁, ?O₂, x}
  have TinS: ?T ⊆ S using OinS and ‹x ∈ S› by auto

  {
    presume ?O₁ ≠ ?O₂ and x ∉ {?O₁, ?O₂}
    then have card ?T = 3 by auto
  }
  then consider
    (primary) card ?T = 3 |
    (limit) x ∈ {?O₁, ?O₂} |
    (degenerate) ?O₁ = ?O₂ by blast
  thus x ∈ cball ?O₁ 1 ∪ cball ?O₂ 1
  proof cases
    case primary
    obtain p and q where p ≠ q and dist p q ≤ 1 and p ∈ ?T and q ∈ ?T
      using property [of ?T] and ‹card ?T = 3› TinS
      by auto
    then have
      dist ?O₁ ?O₂ ≤ 1 ∨ dist ?O₁ x ≤ 1 ∨ dist ?O₂ x ≤ 1
      by (metis dist-commute insertE singletonD)
    thus x ∈ cball ?O₁ 1 ∪ cball ?O₂ 1
      using greatest and TinS
      by fastforce
  next
    case limit
    then have dist x ?O₁ = 0 ∨ dist x ?O₂ = 0 by auto
    thus ?thesis by auto
  next
    case degenerate
    from this greatest TinS have dist ?O₁ x = 0 by auto
```

thus *?thesis* **by** *auto*
qed
qed
qed

Let's make sure that the particular case of points on a plane also works out:

corollary *warmup4*:
fixes S :: (*real* ^ *2*) *set*
assumes *finite* S
assumes *property*: $\bigwedge T.\ T \subseteq S \wedge card\ T = 3 \Longrightarrow \exists p{\in}T.\ \exists q{\in}T.\ p \neq q \wedge dist$ $p\ q \leq 1$
obtains $O_1\ O_2$ **where** $S \subseteq cball\ O_1\ 1 \cup cball\ O_2\ 1$
using *warmup4-generic* **and** *assms* **by** *auto*

end

2 Series I

theory *SeriesI*
imports
Complex-Main
HOL−Analysis.Analysis
begin

2.1 Problem 1

Solve the equation in the integers:

theorem *problem1*:
fixes $x\ y$ :: *int*
assumes $x \neq 0$ **and** $y \neq 0$
shows $1\ /\ x^2 + 1\ /\ (x*y) + 1\ /\ y^2 = 1$
$\longleftrightarrow x = 1 \wedge y = -1 \vee x = -1 \wedge y = 1$
(**is** *?eqn* $\longleftrightarrow$ *?sols*)
proof
— Unfortunately, removing the conversions between int and real takes a few lines
let $?x$ = *real-of-int* x **and** $?y$ = *real-of-int* y
assume *?eqn*
then have $1\ /\ ?x^2 + 1\ /\ (?x*?y) + 1\ /\ ?y^2 = 1$ **by** *auto*
hence $?x^2*?y^2\ /\ ?x^2 + ?x^2*?y^2\ /\ (?x*?y) + ?x^2*?y^2\ /\ ?y^2 = ?x^2*?y^2$
by *algebra*
hence $?x^2 + ?x*?y + ?y^2 = ?x^2 * ?y^2$ **using** ‹$x \neq 0$› ‹$y \neq 0$›
by (*simp add*: *power2-eq-square*)
hence *integ*: $x^2 + x*y + y^2 = x^2 * y^2$
using *of-int-eq-iff* **by** *fastforce*

define g **where** $g = gcd\ x\ y$
then have $g \neq 0$ **and** $g > 0$ **using** ‹$x \neq 0$› ‹$y \neq 0$› **by** *auto*
define $x'\ y'$ **where** $x' = x\ div\ g$ **and** $y' = y\ div\ g$

then have $x' * g = x$ **and** $y' * g = y$ **using** *g-def* **by** *auto*
from *inteq* **and** *this* **have** $g^2 * (x'^2 + x' * y' + y'^2) = x'^2 * y'^2 * g\hat{}4$
by *algebra*
hence *reduced*: $x'^2 + x' * y' + y'^2 = x'^2 * y'^2 * g^2$ **using** $\langle g \neq 0 \rangle$ **by** *algebra*

hence x' *dvd* y'^2 **and** y' *dvd* x'^2
by *algebra+*
moreover have *coprime* x' (y'^2) *coprime* (x'^2) y'
unfolding *x'-def y'-def g-def*
using *assms div-gcd-coprime* **by** *auto*
ultimately have *is-unit* x' *is-unit* y'
unfolding *coprime-def* **by** *auto*
hence *abs1*: $|x'| = 1 \land |y'| = 1$ **using** *assms* **by** *auto*
then consider (*same-sign*) $x' = y'$ | (*diff-sign*) $x' = -y'$ **by** *fastforce*
thus *?sols*
proof *cases*
case *same-sign*
then have $x' * y' = 1$
using *abs1* **and** *zmult-eq-1-iff* **by** *fastforce*
hence $g^2 = 3$
using *abs1 same-sign* **and** *reduced* **by** *algebra*
hence $1^2 < g^2$ **and** $g^2 < 2^2$ **by** *auto*
hence $1 < g$ **and** $g < 2$
using $\langle g > 0 \rangle$ **and** *power2-less-imp-less* **by** *fastforce+*
hence *False* **by** *auto*
thus *?sols* **by** *auto*
next
case *diff-sign*
then have $x' * y' = -1$
using *abs1*
by (*smt mult-cancel-left2 mult-cancel-right2*)
hence $g^2 = 1$
using *abs1 diff-sign* **and** *reduced* **by** *algebra*
hence $g = 1$ **using** $\langle g > 0 \rangle$
by (*smt power2-eq-1-iff*)
hence $x = x'$ **and** $y = y'$
unfolding *x'-def* **and** *y'-def* **by** *auto*
thus *?sols* **using** *abs1* **and** *diff-sign* **by** *auto*
qed
next
assume *?sols*
then show *?eqn* **by** *auto*
qed

2.2 Problem 2

Prove that a sequence is bounded, converges, and find the limit.

context
fixes $a :: real$

```
  assumes a-bounds: 0 < a a < 1
begin
fun c :: nat ⇒ real where
c 0 = a / 2 |
c (Suc n) = (a + (c n)^2) / 2

abbreviation x1 ≡ 1 − sqrt (1 − a)
abbreviation x2 ≡ 1 + sqrt (1 − a)

lemma c-pos: 0 < c n
  using a-bounds
  by (induction n, auto, smt zero-less-power)

lemma c-bounded: c n < x1
proof (induction n)
  case 0
  have (1 − a/2)^2 = 1 − a + (a/2)^2
    by (simp add: power2-diff)
  hence 1 − a < (1 − a/2)^2 using a-bounds by auto
  hence sqrt (1 − a) < 1 − a/2
    using a-bounds and real-less-lsqrt by auto
  thus ?case by auto
next
  case (Suc n)
  then have (c n)^2 < (1 − sqrt (1−a))^2 using c-pos
    by (smt power-less-imp-less-base real-sqrt-abs)
  also have ... = 2 − 2 ∗ sqrt (1−a) − a
    using a-bounds by (simp add: power2-diff)
  finally have (a + (c n)^2)/2 < 1 − sqrt (1−a) by auto
  then show ?case by auto
qed

lemma c-incseq: incseq c
proof (rule incseq-SucI)
  fix n
  from c-bounded have c n < x1 by auto
  have c n < x1 c n < x2
    using c-bounded
    by (smt a-bounds real-sqrt-lt-0-iff)+
  moreover have (c n)^2 − 2∗c n + a = (c n − x1)∗(c n − x2)
    using a-bounds
    by (auto simp add: algebra-simps power2-eq-square)
  ultimately have (c n)^2 − 2∗c n + a > 0
    by (smt nonzero-mult-div-cancel-right zero-le-divide-iff)
  thus c n ≤ c (Suc n) by auto
qed

theorem problem2: c ⟶ x1
proof −
```

```
  obtain L where c ⟶ L
    using c-incseq c-bounded incseq-convergent
    by (metis less-imp-le)
  then have (λn. c (Suc n)) ⟶ L
    using LIMSEQ-Suc by blast
  hence (λn. (a + (c n)^2) / 2 * 2) ⟶ L*2
    using tendsto-mult-right by fastforce
  hence (λn. a + (c n)^2) ⟶ L*2 by (simp del: distrib-right-numeral)
  hence (λn. a + (c n)^2 − a) ⟶ L*2 − a
    using tendsto-diff LIMSEQ-const-iff by blast
  hence (λn. (c n)^2) ⟶ L*2 − a
    by auto
  moreover from ‹c ⟶ L›
  have (λn. (c n)^2) ⟶ L^2
    unfolding power2-eq-square
    using tendsto-mult by blast
  ultimately have L*2 − a = L^2
    by (rule LIMSEQ-unique)
  hence L^2 − 2*L + a = 0 by auto
  moreover have L^2 − 2*L + a = (L − x1)*(L − x2)
    using a-bounds
    by (auto simp add: algebra-simps power2-eq-square)
  ultimately have L = x1 ∨ L = x2
    by auto
  moreover from c-bounded and ‹c ⟶ L› have L ≤ x1
    by (meson LIMSEQ-le-const2 le-less-linear less-imp-triv)
  moreover from a-bounds have x1 < x2 by auto
  ultimately have L = x1 by auto
  thus ?thesis using ‹c ⟶ L› by auto
qed

end

end
```