OM 1969 — Stage 1

Jakub Kądziołka

October 9, 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 $[3\hat{\ }k = (-1{::}int)\hat{\ }k]\ (mod\ 4)$

```
    by (intro cong-pow) (auto simp: cong-def)
  thus ?thesis
    by (auto simp: cong-def minus-one-power-iff)
qed
```

This is, of course, not the only strategy. We leave an alternative proof, in the hope that 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*: *3 powr x = 4∗y + 5 ⟷ x ≥ 0 ∧ even x ∧ y = (3^(nat x) − 5) div 4* **for** *x y :: int*
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
  $(a+b)\hat{\ }4 \leq 8*(a\hat{\ }4 + b\hat{\ }4)$ **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
  $(a+b)\hat{\ }4 \leq 8*(a\hat{\ }4 + b\hat{\ }4)$ **for** $a\ b$ :: *real*
proof –
  have *lemineq*: $2*x\hat{\ }3*y \leq x\hat{\ }4 + x\hat{\ }2*y\hat{\ }2$ **for** $x\ y$ :: *real*
    using *sum-squares-bound* [*of x y*]
      and *mult-left-mono* [**where** $c=x\hat{\ }2$]
    by (*force simp add*: *numeral-eq-Suc algebra-simps*)

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

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

$$
\begin{array}{l}
\textit{convex } S \Longrightarrow \\
\textit{convex-on } S\, f = \\
(\forall k\ u\ x. \\
\quad (\forall i\in\{1..k\}.\ 0 \leq u\ i \wedge x\ i \in S) \wedge \textit{sum } u\ \{1..k\} = 1 \longrightarrow \\
\quad f\ (\sum i = 1..k.\ u\ i *_R x\ i) \leq (\sum i = 1..k.\ u\ i * f\ (x\ i)))
\end{array}
$$

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*:

$$
\begin{array}{l}
\textit{convex-on } s\ f = \\
(\forall x\in s.\ \forall y\in s.\ \forall u\geq 0.\ \forall v\geq 0.\ u + v = 1 \longrightarrow \\
\qquad\qquad\qquad f\ (u *_R x + v *_R y) \leq u * f\, x + v * f\, y)
\end{array}
$$

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 ((λx. x^4) 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 ((λx. 4*x^3) 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 ≤ 12 * x^2 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
  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
  let ?big-pair = arg-max-on ?dist ?pairs
  let ?O₁ = (fst ?big-pair)
  let ?O₂ = (snd ?big-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: ?big-pair ∈ ?pairs by (simp add: arg-max-if-finite)

    have ∀(P,Q)∈?pairs. dist ?O1 ?O2 ≥ dist P Q
      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*
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 $?x = real\text{-}of\text{-}int\ x$ **and** $?y = real\text{-}of\text{-}int\ y$
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** $\langle x \neq 0\rangle\ \langle y \neq 0\rangle$
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*

let $?g = gcd\ x\ y$
let $?x' = x\ div\ ?g$ **and** $?y' = y\ div\ ?g$
have $?g \neq 0$ **and** $?g > 0$ **using** $\langle x \neq 0\rangle\ \langle y \neq 0\rangle$ **by** *auto*
have $?x' * ?g = x$ **and** $?y' * ?g = y$ **by** *auto*
from *integ* **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 $coprime\ ?x'\ (?y'^2)\ coprime\ (?x'^2)\ ?y'$
using *assms div-gcd-coprime* **by** *auto*
ultimately have $is\text{-}unit\ ?x'\ is\text{-}unit\ ?y'$

```
    unfolding coprime-def by auto
  hence abs1: |?x'| = 1 ∧ |?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 ‹?g > 0› 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 ‹?g > 0›
      by (smt power2-eq-1-iff)
    hence x = ?x' and y = ?y' by auto
    thus ?sols using abs1 and diff-sign by auto
  qed
next
  assume ?sols
  then show ?eqn by auto
qed

end
```