Formalizing IMO Problems and Solutions in Isabelle/HOL

Filip Marić
Faculty of Mathematics,
University of Belgrade,
Belgrade, Serbia
filip@matf.bg.ac.rs

Sana Stojanović-Đurđević
Faculty of Mathematics,
University of Belgrade,
Belgrade, Serbia
sana@matf.bg.ac.rs

The *International Mathematical Olympiad (IMO)* is perhaps the most celebrated mental competition in the world and as such is among the greatest grand challenges for Artificial Intelligence (AI). The *IMO Grand Challenge*, recently formulated, requires to build an AI that can win a gold medal in the competition. We present some initial steps that could help to tackle this goal by creating a public repository of mechanically checked solutions of IMO Problems in the interactive theorem prover Isabelle/HOL. This repository is actively maintained by students of the Faculty of Mathematics, University of Belgrade, Serbia within the course ”Introduction to Interactive Theorem Proving”.

1 Introduction

The International Mathematical Olympiad (IMO) is the World Championship Mathematics Competition for High School (pre-college) students and is held annually in a different country. It has gradually expanded to over 100 countries from 5 continents. The competition consists of six problems, which pupils solve during two days (each day they are given three problems and 4.5 hours to work on them). Each problem is worth seven points for the maximum total score of 42 points. The problems are chosen from various areas of secondary school mathematics, generally classifiable as geometry, number theory, algebra, and combinatorics. Problems are expressed with no a priori knowledge of higher mathematics such as calculus and analysis, and solutions are often elementary. Few days before the contest, the International Jury chooses the Olympiad problems. The Jury chooses six problems from the *Shortlist*, a set of around 30 original, beautiful, and difficult problems submitted by mathematicians from around the world. Fully solved shortlists of recent competitions are available on the official IMO web-site (`http://www.imo-official.org/`), while the solutions of all shortlisted problems from 1959-2009 are available in *The IMO Compendium* [1].

IMO is the oldest of International Science Olympiads and is perhaps the most known mental competition in the world. As such it is relevant as one of the greatest grand challenges for Artificial Intelligence (AI). Recently, a group of scientists gathered around the theorem prover Lean, has formulated the *IMO Grand Challenge*[1]: *Build an AI that can win a gold medal in the competition.* To remove ambiguity about the scoring rules, the authors of the challenge propose the formal-to-formal (F2F) variant of the IMO: the AI receives a formal representation of the problem (in the Lean Theorem Prover), and needs to emit a formal (i.e. machine-checkable) proof. A proposal for encoding IMO problems in Lean is currently being developed.

There are already some automated theorem provers capable of solving some specific type of problems. For example, algebraic or semi-algebraic theorem provers are very successful at solving specific classes of geometry problems. However, such provers rarely produce formal proofs and they do not

[1] `http://imo-grand-challenge.github.io/`

J. Marcos, W. Neuper and P. Quaresma (Eds.): Theorem Proving Components for Educational Software 2020 (ThEdu'20)
EPTCS 328, 2020, pp. 35–55, doi:10.4204/EPTCS.328.3

offer, synthetic, human-understandable proofs and justifications (although algebraic proofs can be mechanically checked by analyzing proof certificates [3]). Another example of a successful technique for automated solving of geometry problems is the so-called *Area Method* [4]. Such methods are usually highly specialized for specific classes of problems (e.g., only to some classes of geometric problems), and thus they are very far from general AI.

In this paper we present formalizations of several IMO problem solutions, created within the course "Introduction to interactive theorem proving" at the Faculty of Mathematics, University of Belgrade. Currently problems in algebra, combinatorics and number theory are formalized (geometry problems are skipped, since their formalization requires a rich background theory of high-school synthetic geometry, that is not available in Isabelle/HOL at the moment). All formalized solutions and problem statements (including the three solutions presented in this paper) are publicly available in the GitHub repository `http://github.com/filipmaric/IMO`. We hope that manually constructed proofs and their statements could help better understand the challenges in formalizing IMO solutions and in the long run lead to their better automation.

2 Isabelle/HOL/Isar

In this section we give a brief overview of terms and notions of the Isabelle/Isar proof language used in this paper. This will give the reader only a very rough overview of the syntax used in the rest of the paper, and we refer him to seek more details in the official Isabelle/HOL documentation, preferably [5].

Isabelle/HOL is an incarnation of a simply typed higher-order logic. The type system of Isabelle/HOL is very close to functional programming languages. Base types have the usual names: *bool*, *nat*, *int*, *real*. Type annotations are denoted by ::. For example, 3 :: *int* is an integer constant 3 (numerals are supported by default). Type *nat* is an inductive type of natural numbers in HOL and all values are generated by a constant zero (0) and a constructor (*Suc*). Statements about natural numbers are usually proved by induction. Set types are also supported. The type *'a set* denotes the type of sets with elements of type *'a* (e.g., *nat set* denotes sets of natural numbers). Set of natural numbers less than n, $\{0,1,\ldots,n-1\}$, is denoted by $\{0..{<}n\}$; set of natural numbers less than or equal to n, $\{0,1,\ldots,n\}$, is denoted by $\{0..n\}$; set of natural numbers greater than n is denoted by *{n+1..}*. Function types are denoted by $'a \Rightarrow' b$. For example, function f from integer to real numbers is denoted by $f :: nat \Rightarrow int$.

Terms are built from variables and constants by applying functions and operators. Functions are curried and function application is written in prefix form (as in most functional languages). Terms can also use some advanced mathematical notation. For example $\sum$ *k=0..n. f k* denotes the sum $f(0)+f(1)+\ldots+f(n)$. The same sum can also be denoted by $\sum$ *k≤ n. f k*. Formulae (terms of type *bool*) are built using the usual boolean connectives ($\wedge$, $\vee$, $\neg$, $\longrightarrow$, $\longleftrightarrow$), and quantifiers $\forall x.\ P\ x$ and $\exists x.\ P\ x$. Isabelle/HOL also supports the indefinite description operator *SOME x. P x*, describing any element that satisfies the property P (assuming such element exists) and the definite description operator *THE x. P x* describing the unique element that satisfies the property P (assuming such element exists).

The language Isar is used for writing readable, textbook-like structured theories and proofs. Definitions are introduced by the keyword **definition**, followed by the name of the constant or the function being defined, its type and a defining equality. Recursive functions are defined using the keywords **primrec** and **fun**. Keywords **lemma** and **theorem** are used to state the statement being proved. Statements usually have the form:

lemma
  fixes ⟨*variables*⟩

assumes ⟨*assumptions*⟩
shows ⟨*goal*⟩

Types of variables used in the statement (or in the proof) can be stated upfront with the keyword **fixes** (Isabelle supports type-inference so variables need not always be explicitly declared). Assumptions can be stated after the **assumes** keyword, and conclusions must be stated after the **shows** keyword.

Lemmas and proof rules can also be specified in Isabelle's meta-logic syntax:

$$\bigwedge x_1, \ldots, x_n.\ [\![\ assumption_1, \ldots, assumption_k\ ]\!] \Longrightarrow goal$$

Each lemma and theorem must be followed by a proof. Proofs can be either automatic or interactive, structured proofs.

Automated proofs are specified by a keyword **by** followed by the name of the automated proof method used (most often these are *simp*, *auto*, *force*, *blast*, *metis*, *smt*, *presburger*) and possibly by additional parameters.

Structured proofs in Isar are written within a **proof**-**qed** block. Opening keyword **proof** can be followed by the proof method that is applied in the beginning of the proof. Proofs by case analysis are specified using the method *cases* (e.g., **proof** *(cases "n > 0")*). Proofs by induction are specified using the method *induction* (e.g., **proof** *(induction n rule: less_induct)*). If method is not specified, the system automatically chooses the first method (to be applied). If the proof starts by **proof**-, then no method is being applied.

A typical proof introduces a chain of intermediate statements. Intermediate statements in the proof are given using the keyword **have**, and the final statement (the goal of the current proof) is given using the keyword **show**. This gives the following proof structure.

```
proof-
  have "statement₁" ⟨proof⟩
  have "statement₂" ⟨proof⟩
  ...
  have "statementₖ" ⟨proof⟩
  show ?thesis ⟨proof⟩
qed
```

Here *?thesis* abbreviate the goal of the current proof-block. Each statement introduced by key word **have** or **show** must have its own proof (that, again can be specified either using **by** or **proof**-**qed** block). If automated provers (invoked by the keyword **by**) need to use additional facts, those facts must be explicitly "pumped into the proof context". There are many ways this can be done (for simplicity, we shall use only few). The simplest one is to use the keyword **using** after the statement or the keyword **from** before the statement, followed by the fact that is being inserted into the proof context (and therefore made available to the automated prover). If needed, facts can be named (just beforehand they are stated) and those names can be used instead of explicitly writing the fact enclosed by cartouches ⟨...⟩. If the facts that are used are part of the assumptions of the current lemma, they can be accessed using the abbreviation *assms*.

Usually, intermediate statements are chained and the next statement is proved using the previous one. This gives the following proof structure.

```
proof-
  have "statement₁" ⟨proof⟩
  then have "statement₂" ⟨proof⟩
  ...
```

```
    then show ?thesis ⟨proof⟩
qed
```

Another often encountered proof structure is when several intermediate statements are used to prove the final goal. In Isar, this can be specified using the combination **moreover**-**ultimately**.

```
proof-
    have "statement1" ⟨proof⟩
    moreover have "statement2" ⟨proof⟩
    ...
    moreover have "statementk" ⟨proof⟩
    ultimately show ?thesis ⟨proof⟩
qed
```

Proofs often include chains of equations or inequalities. These are supported in Isar using keywords **also**-**finally**.

```
proof-
    have "t1 = t2" ⟨proof⟩
    also have "... = t3" ⟨proof⟩
    ...
    also have "... = tk" ⟨proof⟩
    finally show ?thesis ⟨proof⟩
qed
```

The last proof, after the keyword **finally** has the fact $t_1 = t_k$ pumped into its proof context. Instead of a chain of equalities, a chain of inequalities or a mixed chain of equalities and inequalities can be used.

3 Example Problems

In this section we shall describe formalizations of three characteristic problems — one in algebra, one in combinatorics and one in number theory.

3.1 A Problem in Algebra

As an example of an algebraic proof that is very straightforward to formalize, we will describe the problem A1 from 2006[2]. Since this proof is very short (both in informal and in formal language), we shall show it in much detail.

We will first present the official solution to this problem, and then we will analyze the formulation of this problem in Isabelle/Isar form:

Problem 1 (2006 A2) *The sequence of real numbers a_0, a_1, a_2, … is defined recursively by*

$$a_0 = -1, \qquad \sum_{k=0}^{n} \frac{a_{n-k}}{k+1} = 0 \quad \textit{for} \quad n \geq 1.$$

Show that $a_n > 0$ for $n \geq 1$.

[2] The problem statement and its solution are described in the official competition bulletin `http://www.imo-official.org/problems/IMO2006SL.pdf`.

Solution. The proof goes by induction. For $n = 1$ the formula yields $a_1 = 1/2$. Take $n \geq 1$, assume $a_1, \ldots, a_n > 0$ and write the recurrence formula for n and $n+1$, respectively as

$$\sum_{k=0}^{n} \frac{a_k}{n-k+1} = 0 \qquad \text{and} \qquad \sum_{k=0}^{n+1} \frac{a_k}{n-k+2} = 0.$$

Subtraction yields

$$0 = (n+2)\sum_{k=0}^{n+1} \frac{a_k}{n-k+2} - (n+1)\sum_{k=0}^{n} \frac{a_k}{n-k+1} = (n+2)a_{n+1} + \sum_{k=0}^{n} \left(\frac{n+2}{n-k+2} - \frac{n+1}{n-k+1} \right) a_k.$$

The coefficient of a_0 vanishes, so

$$a_{n+1} = \frac{1}{n+2} \sum_{k=1}^{n} \left(\frac{n+1}{n-k+1} - \frac{n+2}{n-k+2} \right) a_k = \frac{1}{n+2} \sum_{k=1}^{n} \frac{k}{(n-k+1)(n-k+2)} a_k.$$

The coefficients of $a_1, \ldots, a_n$ are all positive. Therefore, $a_1, \ldots, a_n > 0$ implies $a_{n+1} > 0$.

Writing the statement in the formal language is very straightforward (the sequence a_n is modeled by a function that maps natural indices to real values).

```
theorem IMO_2006_SL_A2:
  fixes a :: "nat ⇒ real"
  assumes "a 0 = -1" "∀ n ≥ 1. (∑ k ≤ n. a (n - k) / (k + 1)) = 0"
  assumes "n ≥ 1"
  shows "a n > 0"
```

Note that the application of the division operator implicitly casts the natural number $k+1$ to real.

In the official solution it is stated that induction is used, but it is not explicitly stated what induction principle is used. A careful examination of the official proof reveals that complete (strong) induction must be used. In the official proof, it is shown that $a_{n+1} > 0$ holds, assuming that $a_k > 0$ holds for all $1 \leq k \leq n$. In Isabelle/HOL, the strong induction principle is given by the rule *less_induct*. All variables are universally quantified.[3]

(⋀ x. (⋀ y. y < x ⟹ P y) ⟹ P x) ⟹ P a

This rule is used in the proof of the theorem.

```
...
  shows "a n > 0"
  using ‹n ≥ 1›
proof (induction n rule: less_induct)
  case (less n)
  show ?case
    ...
qed
```

In formal proof, to show that *a n > 0* holds for arbitrary $n \geq 1$, it suffices to show that *a n > 0* holds for arbitrary $n \geq 1$, under the assumption that for any $1 \leq k < n$ it holds that *a k > 0*. Therefore, many

[3] A more faithful formulation of the *less_induct* rule would be *(⋀ x. (⋀ y. y < x ⟹ ?P y) ⟹ ?P x) ⟹ ?P ?a*. In Isabelle, once the theorem is proved, the object logic variables are lifted into schematic variables. For readability reasons, in this paper we will assume implicit universal quantification.

indices in our formal proof will be shifted by one from the indices in the official, informal proof (since in formal proof we show that *a n* > 0, while informal proof shows that $a_{n+1} > 0$).

The proof then distinguishes the base case ($n = 1$) and the inductive step (when $n > 1$).

```
show ?case
proof cases
  assume n = 1
  . . .
next
  assume n ≠ 1
  . . .
qed
```

The induction base is quite easily discharged. The informal proof just states ”For $n = 1$ the formula yields $a_1 = 1/2$”, and this is quite directly formalized.

```
assume "n = 1"
have "a 1 = 1/2" using assms by auto
then show ?thesis using ‹n = 1› by simp
```

Note that *a 1 = 1/2* follows automatically from the theorem assumptions ($a_0 = -1$ and $\sum_{k=0}^{n} \frac{a_{n-k}}{k+1} = 0$, denoted by *assms*), and this shows the goal *a n* > 0 (denoted by *?thesis*), since in this case it holds that $n = 1$.

As usual, the inductive step is much harder. The informal proof goes as follows. ”Take $n \geq 1$, assume $a_1, \ldots, a_n > 0$ and write the recurrence formula for n and $n+1$, respectively as

$$\sum_{k=0}^{n} \frac{a_k}{n-k+1} = 0 \qquad \text{and} \qquad \sum_{k=0}^{n+1} \frac{a_k}{n-k+2} = 0.$$

Subtraction yields

$$0 = (n+2) \sum_{k=0}^{n+1} \frac{a_k}{n-k+2} - (n+1) \sum_{k=0}^{n} \frac{a_k}{n-k+1}.”$$

Formalization follows this quite faithfully (except that indices are shifted by one).

```
have "(∑ k < n. a k / (n - k)) = 0"
  using assms(2)[of "n - 1"] ‹n > 1› sum.nat_diff_reindex[of "λ k. a k / (n - k)" "n"]
  by simp
moreover have "(∑ k < n + 1. a k / (n + 1 - k)) = 0"
  using assms(2)[of "n"] ‹n > 1› sum.nat_diff_reindex[of "λ k. a k / (n + 1 - k)" "n + 1"]
  by simp
ultimately have "(n + 1) * (∑ k < n + 1. a k / (n + 1 - k)) - n * (∑ k < n. a k / (n - k)) = 0"
  by simp
```

The proof steps use the lemma *sum.nat_diff_reindex*, already available in Isabelle/HOL:

```
(∑ i < n. g (n - Suc i)) = (∑ i < n. g i)
```

It's use in the informal proof is only implicit (sum re-indexing is considered fully trivial).

The informal proof continues by giving the equality between the difference of two sums with the expression:

$$(n+2)a_{n+1} + \sum_{k=0}^{n} \left(\frac{n+2}{n-k+2} - \frac{n+1}{n-k+1} \right) a_k.$$

This is also done in the formal proof, except that indices are again off by one.

```
then have "(n + 1) * a n = - (∑ k < n. ((n + 1) / (n + 1 - k) - n / (n - k)) * a k)"
  by (simp add: algebra_simps sum_distrib_left sum_subtractf)
then have "(n + 1) * a n = (∑ k < n. (n / (n - k) - (n + 1) / (n + 1 - k)) * a k)"
  by (simp add: algebra_simps sum_negf[symmetric])
```

The first proof step uses various algebraic simplifications (contained in the collection of theorems called *algebra_simps*) and the following two properties of sums (*sum_distrib_left* and *sum_subtractf*):

```
r * (∑ n ∈ A. f n) = (∑n∈A. r * f n)          (∑x∈A. f x - g x) = (∑x∈A. f x) - (∑x∈A. g x)
```

The second proof step uses various algebraic simplifications (collection of theorems *algebra_simps*) and the following property of sums (*sum_negf*):

```
(∑x∈A. - f x) = - (∑x∈A. f x)
```

Note that although this formal proof is essentially the same as the informal proof, it is a bit more verbose. Some intermediate steps had to be specified and proved using several lemmas already available in Isabelle/HOL. Without the use of intermediate steps, automated provers in Isabelle were not able to directly prove the final goal.

The informal proof continues: "The coefficient of a_0 vanishes, so

$$a_{n+1} = \frac{1}{n+2} \sum_{k=1}^{n} \left(\frac{n+1}{n-k+1} - \frac{n+2}{n-k+2} \right) a_k = \frac{1}{n+2} \sum_{k=1}^{n} \frac{k}{(n-k+1)(n-k+2)} a_k.$$

The coefficients of $a_1, \dots, a_n$ are all positive. Therefore, $a_1, \dots, a_n > 0$ implies $a_{n+1} > 0$".

The formal proof roughly follows this.

```
also have "... = (∑ k ∈ {1..<n}. (n / (n - k) - (n + 1) / (n + 1 - k)) * a k)"
  using ‹n > 1›
  by (subst sum_remove_zero, auto)
```

It is easily automatically deduced that the coefficient of a_0 is $n/(n-0) - (n+1)/(n+1-0) = 0$. However, it was necessary to formulate, and to prove, a separate lemma for isolating the first member of the sum.

```
lemma sum_remove_zero:
  fixes n :: nat
  assumes "n > 0"
  shows "(∑ k < n. f k) = f 0 + (∑ k ∈ {1..<n}. f k)"
  using assms
  by (simp add: atLeast1_lessThan_eq_remove0 sum.remove)
```

Next, in the informal proof it trivially holds that the sum is positive, since the coefficients of $a_1, \ldots, a_n$ are all positive. But in the formal proof it is proved that the sum is positive by proving that all its members are positive. This is given by the Isabelle/HOL rule *sum_pos*.

$$\llbracket \mathit{finite}\ I;\ I \neq \{\};\ \bigwedge i.\ i \in I \Longrightarrow 0 < f\, i \rrbracket \Longrightarrow 0 < (\sum x \in I.\ f\, x)$$

Applying this rule requires proving it's three assumptions (separated by the word *next*).

```
also have "... > 0"
proof (rule sum_pos)
  show "finite {1..<n}" by simp
next
  show "{1..<n} ≠ {}" using ‹n > 1› by simp
next
  fix i
  assume "i ∈ {1..<n}"
  show "(n / (n - i) - (n + 1) / (n + 1 - i)) * a i > 0" (is "?ci * a i > 0")
  proof-
    have "a i > 0" using less ‹i ∈ {1..<n} › by simp
    moreover have "?ci > 0"
    proof-
      have "?ci = i / ((n - i) * (n + 1 - i))" using ‹i ∈ {1..<n}› by (simp add: field_simps of_nat_diff)
      then show ?thesis using ‹i ∈ {1..<n}› by simp
    qed
    ultimately show ?thesis by simp
qed
```

The only non-trivial part in this proof is proving that the coefficient *?ci* of *a i* is non-negative (the first step of the third assumption). This is done essentially in the same way as in the informal proof — two fractions are subtracted, and reduced to a common fraction with a numerator and denominator that are obviously positive. To show that each *a i* is positive, the induction hypothesis (denoted by *less*) is used.

The proof finishes by noting that we have proved that *(n+1) · (a n)* is positive, and that, since *n+1* is positive, so must also be *a n* (the last proof is found by the Sledgehammer tool and uses an SMT solver). This step is implicit in informal proof.

```
finally have "(n + 1) * (a n) > 0" .
then show ?thesis by (smt mult_nonneg_nonpos of_nat_0_le_iff)
```

3.2 A Problem in Combinatorics

As an example of a problem that has a very short and elegant informal solution, but which is hard to formalize we show the problem C1 from 2017[4].

Problem 2 (2017 C1) *A rectangle $\mathcal{R}$ with odd integer side lengths is divided into small rectangles with integer side lengths. Prove that there is at least one among the small rectangles whose distances from the four sides of $\mathcal{R}$ are either all odd or all even.*

[4] The problem statement and its solution are described in the official competition bulletin `http://www.imo-official.org/problems/IMO2017SL.pdf`.

The first major challenge is to give a formal statement of the problem. Although at first glance one might think that division must form a rectangular grid (as shown on the left picture in Figure 1), more general tilings are allowed (as shown on the right picture in Figure 1).

We shall assume that a coordinate system is introduced so that the lower left corner of the big rectangle is in its origin. Each rectangle will be determined by four non-negative integers (x_1, x_2, y_1, y_2): coordinates of its left, right, bottom and top line. Unit squares are indicated by checkerboard pattern, so our big rectangle can be represented by the quadruple $(0, 17, 0, 11)$. For a rectangle to be valid (non-empty) it must hold that $x_1 < x_2$ and that $y_1 < y_2$.

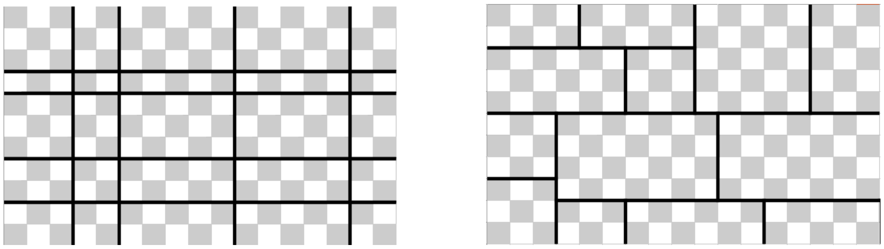

Figure 1: Tiling of a rectangle

We formalize this as follows (since all coordinates are positive, instead of integers we use natural numbers). The validity condition could have been encoded in the rectangle type, but that would require using a bit more advanced features of Isabelle/HOL, so we did not go in that direction.

```
type_synonym rect = "nat × nat × nat × nat"

fun valid_rect :: "rect ⇒ bool" where "valid_rect (x1, x2, y1, y2) ⟷ x1 < x2 ∧ y1 < y2"
```

Each unit square in a rectangle is characterized by its two integer coordinates (of its lower left corner). Each valid rectangle contains a set of unit squares that can be obtained by their coordinates as a Cartesian product of two discrete integer intervals.

```
type_synonym square = "nat × nat"
fun squares :: "rect ⇒ square set" where "squares (x1, x2, y1, y2) = {x1..<x2} × {y1..<y2}"
```

We define a tiling (a subdivision) of a rectangle $\mathscr{R}$ to be a set of non-overlapping rectangles that cover $\mathscr{R}$. Two rectangles overlap if they share a common square. A set of rectangles is non-overlapping if no two different rectangles overlap. A set of rectangles cover a given rectangle $\mathscr{R}$ if the union of all squares is equal to the set of squares of $\mathscr{R}$. We formalize this as follows.

```
definition overlap :: "rect ⇒ rect ⇒ bool" where "overlap r1 r2 ⟷ squares r1 ∩ squares r2 ≠ {}"

definition non_overlapping :: "rect set ⇒ bool" where
  "non_overlapping rs ⟷ (∀ r1 ∈ rs. ∀ r2 ∈ rs. r1 ≠ r2 ⟶ ¬ overlap r1 r2)"

definition cover :: "rect set ⇒ rect ⇒ bool" where "cover rs r ⟷ (⋃ (squares ‘ rs)) = squares r"

definition tiles :: "rect set ⇒ rect ⇒ bool" where "tiles rs r ⟷ cover rs r ∧ non_overlapping rs"
```

Finally, we can give the formal statement of the theorem.

```
theorem IMO_2017_SL_C1:
  fixes a b :: nat
  assumes "odd a" "odd b" "tiles rs (0, a, 0, b)" "∀ r ∈ rs. valid_rect r"
  shows "∃ (x1, x2, y1, y2) ∈ rs. let ds = {x1 - 0, a - x2, y1 - 0, b - y2}
                                  in (∀ d ∈ ds. even d) ∨ (∀ d ∈ ds. odd d)"
```

The informal proof is very short and elegant.

”Let the width and height of $\mathscr{R}$ be odd numbers a and b. Divide $\mathscr{R}$ into ab unit squares and color them green and yellow in a checkered pattern. Since the side lengths of a and b are odd, the corner squares of $\mathscr{R}$ will all have the same color, say green. Call a rectangle (either $\mathscr{R}$ or a small rectangle) green if its corners are all green; call it yellow if the corners are all yellow, and call it mixed if it has both green and yellow corners. In particular, $\mathscr{R}$ is a green rectangle.”

Several definitions are introduced to formalize this passage, and one simple lemma is proved.

```
fun green :: "square ⇒ bool" where "green (x, y) ⟷ (x + y) mod 2 = 0"
fun yellow :: "square ⇒ bool" where "yellow (x, y) ⟷ (x + y) mod 2 ≠ 0"

fun corners :: "rect ⇒ square set" where
  "corners (x1, x2, y1, y2) = {(x1, y1), (x1, y2-1), (x2-1, y1), (x2-1, y2-1)}"

definition green_rect :: "rect ⇒ bool" where "green_rect r ⟷ (∀ c ∈ corners r. green c)"
definition yellow_rect :: "rect ⇒ bool" where "yellow_rect r ⟷ (∀ c ∈ corners r. yellow c)"
definition mixed_rect :: "rect ⇒ bool" where "mixed_rect r ⟷ ¬ green_rect r ∧ ¬ yellow_rect r"

lemma
  assumes "odd a" "odd b"
  shows "green_rect (0, a, 0, b)"
  unfolding green_rect_def by auto
```

The informal proof continues as follows. ”We will use the following trivial observations.

- Every mixed rectangle contains the same number of green and yellow squares;
- Every green rectangle contains one more green square than yellow square;
- Every yellow rectangle contains one more yellow square than green square.”

This is where things start to get harder. Unfortunately, these observations seem far from trivial, when it comes to their formal proofs. Giving their formal statement requires following two definitions.

```
definition green_squares :: "rect ⇒ square set" where
  "green_squares r = {(x, y) ∈ squares r. green (x, y)}"
definition yellow_squares :: "rect ⇒ square set" where
  "yellow_squares r = {(x, y) ∈ squares r. yellow (x, y)}"
```

Stating the observations is now straightforward (we only show the middle one).

```
lemma green_rect:
  assumes "valid_rect (x1, x2, y1, y2)" "green_rect (x1, x2, y1, y2)"
  shows "card (green_squares (x1, x2, y1, y2)) = card (yellow_squares (x1, x2, y1, y2)) + 1"
```

There are two main approaches to prove this lemma. The first approach requires explicitly calculating the number of green and the number of yellow squares in a green rectangle. The other approach requires to establish a bijective mapping between all yellow squares and all but one green square in a rectangle. We have taken the first approach. The number of green and yellow squares in a rectangle depends on the rectangle dimensions, but also on whether the first square (the one with the coordinates $(0,0)$) is a green or yellow square. We will divide the set of squares (in a rectangle) into two halves in the following manner. Denote the total number of squares by k. If k is even, those halves will be equal, and if k is odd, one half will be greater by one square than the other half. In both cases, the number of squares of the half that contains the starting square is equal $k/2$ rounded upwards (by the ceiling function) and the number of squares of the other half is equal $k/2$ rounded downwards (by the floor function). This is formalized by the following lemmas (the expression *k* div *2* rounds $k/2$ downwards, and *(k + 1)* div *2* rounds it upwards).

lemma
  assumes *"green* (x_1, y_1)*" "valid_rect* (x_1, x_2, y_1, y_2)*"*
  shows *"card (yellow_squares* $(x_1, x_2, y_1, y_2)) = ((x_2 - x_1) * (y_2 - y_1))$ div *2"*
    "card (green_squares $(x_1, x_2, y_1, y_2)) = ((x_2 - x_1) * (y_2 - y_1) + 1)$ div *2*

Only the first part of this lemma (the number of green squares) needed to be explicitly proved, while the number of yellow squares is easily calculated as the difference between the total number of squares and the number of green squares. An analogous lemma for a yellow starting square (x_1, y_1) is proved (again not directly, but by reducing it to the present lemma for a green starting square, by translating the whole rectangle by one square to the right).

The correctness of the formula for the number of green squares of a rectangle that starts on a green square is proved by induction on the height of a rectangle $(y_2 - y_1 - 1)$. Both in the base case and in the inductive step, a lemma that characterizes the number of green squares in a single row is used (it covers both the case of a green and a yellow starting square in that row).

lemma
  assumes *"x1 < x2"*
  shows *"card* $\{(x, y).\ x1 \leq x \wedge x < x2 \wedge y = y0 \wedge green\ (x, y)\}$ *=*
    *(*if *green (x1, y0)* then *(x2 - x1 + 1)* div *2* else *(x2 - x1)* div *2)"*

This lemma is also proved by mathematical induction, this time over the length of the row $(x_2 - x_1 - 1)$. In both inductive proofs the set of squares of the rectangle is given as a disjoint union of squares of a smaller rectangle (for which we know the number of green squares by induction hypothesis) and a degenerated rectangle (a row, i.e., a single square) for which we directly calculate the number of green squares (depending on the color of its first square).

With those lemmas in place, the number of green and yellow squares in a green rectangle can easily be connected, by also noting that both dimensions of that rectangle must be odd, so its total number of squares is odd. The case of a yellow rectangle is fully analogous, while the mixed rectangle must have an even number of squares that is evenly split between green and yellow squares.

The informal proof continues as follows. ”The rectangle $\mathscr{R}$ is green, so it contains more green unit squares than yellow unit squares. Therefore, among the small retangles, at least one is green.”

This is formalized by the following theorem (proved for any green rect).

lemma
  assumes *"green_rect* (x_1, x_2, y_1, y_2)*" "valid_rect* (x_1, x_2, y_1, y_2)*"*
    "tiles rs (x_1, x_2, y_1, y_2)*"* *"*$\forall$ *r* $\in$ *rs. valid_rect r"*
  shows *"*$\exists$ *r* $\in$ *rs. green_rect r"*

The proof is by contradiction (we will show only the proof outline). If the negation of the thesis is assumed, then all tiles are either yellow or mixed, so, by the previous lemmas, they have less or equal green than yellow squares.

```
proof (rule ccontr)
assume "¬ ?thesis"
then have "∀ r ∈ rs. yellow_rect r ∨ mixed_rect r" using mixed_rect_def
  ⟨proof⟩
then have "∀ r ∈ rs. card (green_squares r) ≤ card (yellow_squares r)"
  ⟨proof⟩
```

Therefore, the total number of green squares in the big rectangle is less or equal to the number of yellow squares in the big rectangle, which is contradictory to earlier lemma about green rectangles. This proof goes as follows.

```
have "card (green_squares (x1, x2, y1, y2)) ≤ card (yellow_squares (x1, x2, y1, y2))"
proof-
  have "card (green_squares (x1, x2, y1, y2)) = card (⋃ (green_squares ‘ rs))" ⟨proof⟩
  also have "... = (∑ r ∈ rs. card (green_squares r))" ⟨proof⟩
  also have "... ≤ (∑ r ∈ rs. card (yellow_squares r))" ⟨proof⟩
  also have "... = card (⋃ (yellow_squares ‘ rs))" ⟨proof⟩
  also have "... = card (yellow_squares (x1, x2, y1, y2))" ⟨proof⟩
  finally show ?thesis .
qed
then show False using ‘green_rect (x1, x2, y1, y2)‘ ‘valid_rect (x1, x2, y1, y2)‘ green_rect by auto
```

Showing that the cardinality of the union is the sum of cardinalities of its members is not trivial, since it requires proving that the union is disjoint and that all involved sets are finite. For example, the outline of the second subproof in the previous proof is the following.

```
have "card (⋃ (green_squares ‘ rs)) = (∑ r ∈ rs. card (green_squares r))"
proof (rule card_UN_disjoint)
  show "finite rs" ⟨proof⟩
  show "∀ r ∈ rs. finite (green_squares r)" by auto
  show "∀ r1 ∈ rs. ∀ r2 ∈ rs. r1 ≠ r2 ⟶ green_squares r1 ∩ green_squares r2 = {}" ⟨proof⟩
qed
```

To show that the tiling *rs* must contain a finite number of rectangles we show that each tile must be inside the big rectangle, and that there are only finitely many rectangles that can be inside a given rectangle (*rs* is the subset of $\{x_1..x_2\} \times \{x_1..x_2\} \times \{y_1..y_2\} \times \{y_1..y_2\}$, which is finite). The disjointness of sets of green squares follows from the fact that the tiles are non-overlapping.

The informal proof finishes as follows. ”Let $\mathscr{S}$ be such a small green rectangle, and let its distances from the sides of $\mathscr{R}$ be x, y, u and v. The top-left corner of $\mathscr{R}$ and the top-left corner of $\mathscr{S}$ have the same color, which happens if and only if x and u have the same parity. Similarly, the other three green corners of $\mathscr{S}$ indicate that x and v have the same parity, y and u have the same parity, i.e. x, y, u and v are all odd or all even.”

This is formalized as follows.

```
definition inside :: "rect ⇒ rect ⇒ bool" where "inside ri ro ⟷ squares ri ⊆ squares ro"

lemma
  assumes "valid_rect (x^i_1, x^i_2, y^i_1, y^i_2)" "green_rect (x^i_1, x^i_2, y^i_1, y^i_2)" "green_rect (x^o_1, x^o_2, y^o_1, y^o_2)"
          "inside (x^i_1, x^i_2, y^i_1, y^i_2) (x^o_1, x^o_2, y^o_1, y^o_2)"
  shows "let ds = {x^i_1 - x^o_1, x^o_2 - x^i_2, y^i_1 - y^o_1, y^o_2 - y^i_2} in (∀ d ∈ ds. even d) ∨ (∀ d ∈ ds. odd d)"
```

Interestingly, this lemma can be proved almost fully automatically. Even though its informal proof is given in more detail than previous proofs that required much longer formal proofs.

3.3 A Problem in Number Theory

As an example of a problem in number theory we show the formalization of the problem N1 from 2017[5]. The formalization follows the official solutions, but reveals many gaps typical for informal proofs.

Problem 3 (2017 N1) *The sequence $a_0, a_1, a_2, \ldots$ of positive integers satisfies*

$$a_{n+1} = \begin{cases} \sqrt{a_n}, & \text{if } \sqrt{a_n} \text{ is an integer} \\ a_n + 3, & \text{otherwise} \end{cases} \qquad \text{for every } n \geq 0$$

Determine all values of $a_0 > 1$ for which there is at least one number a such that $a_n = a$ for infinitely many values of n.

The answer is ”all positive multiples of 3” and we can easily formalize the problem statement. A slight problem is that *sqrt* in Isabelle/HOL is defined only for real numbers, so to avoid using reals we define the square root of naturals.

definition *sqrt_nat :: "nat ⇒ nat"* **where**
  *"sqrt_nat x = (THE s. x = s * s)"*

theorem *IMO_2017_SL_N1:*
  fixes *a :: "nat ⇒ nat"*
  assumes *"∀ n. a (n + 1) = (if (∃ s. a n = s * s) then sqrt_nat (a n) else (a n) + 3)"* **and** *"a 0 > 1"*
  shows *"(∃ A. infinite n. a n = A) ⟷ a 0 mod 3 = 0"*

The informal proof begins as follows. ”Since the value of a_{n+1} only depends on the value of a_n, if $a_n = a_m$ for two different indices n and m, then the sequence is eventually periodic. So we look for the values of a_0 for which the sequence is eventually periodic.” This is formulated by the following definition and a series of lemmas.

definition *eventually_periodic :: "(nat ⇒ ’a) ⇒ bool"* **where**
  "eventually_periodic a ⟷ (∃ p > 0. ∃ n_0. ∀ n ≥ n_0. a (n + p) = a n)"

lemma
  fixes *a :: "nat ⇒ ’a"*
  assumes *"∀ n ≥ n_0. a (n + p) = a n"*
  shows *"∀ k. a (n_0 + k * p) = a n_0"*
⟨*proof*⟩

lemma
  fixes *a :: "nat ⇒ ’a"*
  assumes *"∀ n. a (n + 1) = f (a n)" "a n_1 = a n_2"*
  shows *"∀ k. a (n_1 + k) = a (n_2 + k)"*
⟨*proof*⟩

lemma

[5] The problem statement and its solution are available in the official competition bulletin `http://www.imo-official.org/problems/IMO2017SL.pdf`.

fixes *a :: "nat ⇒ ’a"*
assumes *"∀ n. a (n + 1) = f (a n)" "n_1 < n_2" "a n_1 = a n_2"*
shows *"eventually_periodic a"*
⟨*proof*⟩

lemma
fixes *a :: "nat ⇒ ’a"*
assumes *"∀ n. a (n + 1) = f (a n)"*
shows *"(∃ A. infinite {n. a n = A}) ⟷ eventually_periodic a"*
⟨*proof*⟩

The first two lemmas are proved by induction (on the value k). The third lemma is proved by applying the second lemma to prove that the sequence periodically repeats after $n_2 - n_1$ elements, starting on the index n_1. The proof of the last lemma is split into two directions. In the first direction, we assume that there exists an infinite set of indexes that contains elements with the same value. In that case, there must be two different values n_1 and n_2 such that $a\ n_1 = a\ n_2$, so the sequence is eventually periodic by the third lemma. In the second direction of the proof, we assume that the sequence is periodic, then by the first lemma, the sequence attains the value a_{n_0} on the infinite set of indices $n_0 + k \cdot p$.

The informal proof then continues by a series of claims and their proofs.

Claim 1. ”If $a_n \equiv -1 \pmod 3$, then, for all $m > n$, a_m is not a perfect square. It follows that the sequence is eventually strictly increasing, so it is not eventually periodic.
Proof. A square cannot be congruent to -1 modulo 3, so $a_n \equiv -1 \pmod 3$ implies that a_n is not a square, therefore $a_{n+1} = a_n + 3 > a_n$. As a consequence, $a_{n+1} \equiv a_n \equiv -1 \pmod 3$, so a_{n+1} is not a square either. By repeating the argument, we prove that, from a_n on, all terms of the sequence are not perfect squares and are greater than their predecessors, which completes the proof.”

First we formalize the notion of being eventually increasing, give its equivalent characterization (proved by induction) and prove that a strictly increasing sequence cannot be periodic (the proof by contradiction is very simple since eventually strictly increasing and periodic sequence would have two values a_n and a_{n+p} for which it would have to hold both $a_n = a_{n+p}$ and $a_n < a_{n+p}$).

definition *eventually_increasing :: "(nat ⇒ nat) ⇒ bool"* **where**
"eventually_increasing a ⟷ (∃ n_0. ∀ n ≥ n_0. a n < a (n + 1))"

lemma
shows *"eventually_increasing a ⟷ (∃ n_0. ∀ i j. n_0 ≤ i ∧ i < j ⟶ a i < a j)"*
⟨*proof*⟩

lemma
assumes *"eventually_increasing a"*
shows *"¬ eventually_periodic a"*
⟨*proof*⟩

We need to formulate and prove that ”A square cannot be congruent to -1 modulo 3”. There is no need to use negative numbers since $a_n \equiv -1 \pmod 3$ is equivalent to $a_n \equiv 2 \pmod 3$.

lemma
fixes *s :: nat*
shows *"(s * s) mod 3 ≠ 2"*
⟨*proof*⟩

Claim1 establishes several facts, but the only ”takeaway”, i.e., the only fact that is used later in the proof is that if $a_n \equiv -1 \pmod 3$, then the sequence is not eventually periodic. Since *Claim1* is useful only for the proof of the main theorem, we do not formulate it as a general lemma, but instead we formulate it as a named intermediate fact within the proof of the main theorem:

have *Claim1: "$\exists$ n. a n mod 3 = 2 $\Longrightarrow$ $\neg$ eventually_periodic a"*

The informal proof gives a delicate connection between the fact that elements are not full squares and that the sequence is strictly increasing. The language construction ”by repeating the argument” indicates that the proof is essentially based on mathematical induction. Therefore, to prove the previous claim we prove the following statement (by induction on the value $m-n$).

have *"$\forall$ m $\geq$ n. ($\nexists$ s. a m = s * s) $\wedge$ a m mod 3 = 2 $\wedge$ a (m + 1) = a m + 3"* ⟨*proof*⟩

From this it easily follows that the sequence is eventually increasing, and therefore, by a previous lemma, not eventually periodic.

The informal proof continues with a second claim.

Claim 2. ”If $a_n \not\equiv -1 \pmod 3$ and $a_n > 9$ then there is an index $m > n$ such that $a_m < a_n$.
Proof. Let t^2 be the largest perfect square which is less than a_n. Since $a_n > 9$, t is at least 3. The first square in the sequence $a_n, a_n+3, a_n+6, \ldots$ will be $(t+1)^2, (t+2)^2, (t+3)^2$, therefore there is an index $m > n$ such that $a_m \leq t+3 < t^2 < a_n$, as claimed.”

This claim is very easily stated (again as a named intermediate fact within the main proof).

have *Claim2: "$\forall$ n. a n mod 3 $\neq$ 2 $\wedge$ a n > 9 $\longrightarrow$ ($\exists$ m > n. a m < a n)"*

However, formal proof of this claim is very involved, since the informal proof is very imprecise. First we define the value of *?t*, and prove its basic properties. It requires showing that the set *?T* of all perfect squares less than *a n* is finite and non-empty (it contains the number 3 so it must be non-empty).

let *?T = "{t | t. t*t < a n}"* **and** *?t = "Max ?T"*
have *"?t $\geq$ 3" "?t^2 < a n" "a n $\leq$ (?t + 1)^2"* ⟨*proof*⟩

The claim ”The first square in the sequence $a_n, a_n+3, a_n+6, \ldots$ will be $(t+1)^2, (t+2)^2, (t+3)^2$” was very hard to prove formally. The statement is formalized as follows.

have *"$\exists$ k. a (n + k) $\in$ {(?t+1)^2, (?t+2)^2, (?t+3)^2}"*

Note that we used a_{n+k} instead of $a_n + 3k$. Although that is essentially the same, since these two sequences coincide until a perfect square occurs, the sequences $a_n, a_n+3, a_n+6, \ldots$ and $a_n, a_{n+1}, a_{n+2}, \ldots$ must be formally linked (and this is going to be done within the proof). To prove the given statement, we first prove the following number-theoretic fact.

have *"a n mod 3 = (?t+1)^2 mod 3 $\vee$ a n mod 3 = (?t+2)^2 mod 3 $\vee$ a n mod 3 = (?t+3)^2 mod 3"*

Since it is assumed that *a n mod 3* is not 2, it must be either 0 or 1. The proof of the fact then follows from the next general lemma (whose proof goes by a case analysis of the value of *t mod 3*).

lemma
  fixes *t :: nat*
  shows *"{(t + 1)^2 mod 3, (t + 2)^2 mod 3, (t + 3)^2 mod 3} = {0, 1}"*
⟨*proof*⟩

The sought value *a (n+k)* that belongs to the set $\{(?t{+}1)^2,\ (?t{+}2)^2,\ (?t{+}3)^2\}$ will be equal to the first element of the sequence $(?t{+}1)^2,\ (?t{+}2)^3,\ (?t{+}3)^2$ that is congruent to *a n* modulo 3. We prove this in the form of the following auxiliary claim.

fix *i*
assume $"i > 0"$ **and** $"\forall\, i'.\ 0 < i' \wedge i' < i \longrightarrow a\ n\ mod\ 3 \neq (?t + i')^2\ mod\ 3"$ **and**
$"a\ n\ mod\ 3 = (?t + i)^2\ mod\ 3"$
have $"\exists\, k.\ a\ (n + k) = (?t + i)^2"$
⟨*proof*⟩

The sought index $?k$ is $((?t + i)^2 - a\ n)\ div\ 3$. We prove that $a_{n+?k}$ will be equal to $(?t + i)^2$, and that it will be the first perfect square in the sequence $a_n, a_{n+1}, \ldots$ This follows from the following fact:

have $"\forall\, k' \leq\ ?k.\ a\ (n + k') = a\ n + 3 * k'"$

The proof goes by induction on k'. The base case is trivial. In the inductive step we need to show that the equality holds for $k'+1 \leq ?k$ under the assumption that it holds for $k' < ?k$. It suffices to prove that *a (n+k')* is not a full square. We prove that by contradiction. If it were a full square, since $k'{+}1 \leq\ ?k = ((?t + i)^2 - a\ n)\ div\ 3$ it would hold that $3 * (k' + 1) \leq (?t + i)^2 - a\ n$, i.e., that $a\ (n{+}k') = a\ n + 3 * k' \leq (?t + i)^2 - 3$. Therefore, $a(n+k')$ would be a full square strictly between $?t^2$ and $(?t+i)^2$, which is impossible by our assumption that $\forall\, i'.\ 0 < i' \wedge i' < i \longrightarrow a\ n\ mod\ 3 \neq (?t + i')^2\ mod\ 3$. Since *a (n+k')* is not a full square, it holds that $a\ (n{+}k'{+}1) = a\ (n{+}k') + 3 = a\ n + 3{*}k + 3 = a\ n + 3{*}(k{+}1)$. This finishes the inductive proof.
Therefore, $a\ (n +\ ?k) = a\ (n + 3{*}?k)$ and $a\ (n + 3{*}?k) = (?t + i)^2$ (this holds directly by the definition of $?k = ((?t + i)^2 - a\ n)\ div\ 3$). So there indeed exists k such that $a\ (n + k) = (?t + i)^2$, which finishes the proof of the axuiliary claim.
The statement $\exists\, k.\ a\ (n + k) \in \{(?t{+}1)^2, (?t{+}2)^2, (?t{+}3)^2\}$ is then proved by case analysis of the fact $a\ n\ mod\ 3 = (?t{+}1)^2\ mod\ 3 \vee a\ n\ mod\ 3 = (?t{+}2)^2\ mod\ 3 \vee a\ n\ mod\ 3 = (?t{+}3)^2\ mod\ 3$, applying the auxiliary claim for $i = 1$, $i = 2$, and $i = 3$. From that it easily follows that $\exists\, k.\ a\ (n + k + 1) \in \{?t{+}1, ?t{+}2, ?t{+}3\}$, so $a\ (n + k + 1) \leq\ ?t + 3 <\ ?t^2 < a\ n$, finishing he formal proof of *Claim2*.

The next claim in the informal proof is the following.

Claim 3. "If $a_n \equiv 0 \pmod 3$, then there is an index $m > n$ such that $a_m = 3$.
Proof. First we notice that, by the definition of the sequence, a multiple of 3 is always followed by another multiple of 3. If $a_n \in \{3,6,9\}$ the sequence will eventually follow the periodic pattern $3,6,9,3,6,9,\ldots$. If $a_n > 9$, let j be an index such that a_j is equal to the minimum value of the set $\{a_{n+1}, a_{n+2}, \ldots\}$. We must have $a_j \leq 9$, otherwise we could apply *Claim2* to a_j and get a contradiction on the minimality hypothesis. It follows that $a_j \in \{3,6,9\}$, and the proof is complete."

By analyzing the informal proof we note that the result for the case when $a_n \leq 9$ is also applied within the case $a_n > 9$. Therefore, it is wise to prove that case as the following sub-claim.

have *Claim3_a:* $"\forall\, n.\ a\ n\ mod\ 3 = 0 \wedge a\ n \leq 9 \longrightarrow (\exists\, m > n.\ a\ m = 3)"$

It is easy to prove by using the recursive definition of *a* and direct calculations, except one little detail. If $a\ n\ mod\ 3 = 0$, and $a\ n \leq 9$ then n could be 0, 3, 6, or 9. In the case 3, 6, and 9, the claim then easily follows by direct calculations using the definition of the sequence *a*. This is not the case if $a\ n = 0$. To prove that this case is impossible, we must again use induction to prove.

have $"\forall\, n.\ a\ n > 1"$

This proof uses the hypothesis *a 0 > 1*, and the fact that the square root of a number that is strictly greater than 1 must itself be a number strictly greater than 1.

*C*laim3 is then stated as follows:

have *Claim3: "$\forall$ n. a n mod 3 = 0 $\longrightarrow$ ($\exists$ m > n. a m = 3)"*

Its proof starts by case analysis on whether *a n $\leq$ 9*. The case when *a n $\leq$ 9* is already covered by the sub-claim. If *a n > 9*, we follow the informal proof by taking the minimal element of the set *a ‘ {n+1..}* (it is the image of the set *{n+1..}* under the function *a*). Unfortunately, we cannot use the operator *Min* to find its minimum, since it is intended only for finite sets. An arbitrary infinite set does not need to have a minimal element. However, this is the set of natural numbers, and one of the characteristic features of natural numbers is that they are well-ordered i.e., that each non-empty set contains a minimal element. Isabelle/HOL supports the binder *LEAST* that determines the least element in a well-ordered set and we will use it to define the value ?*m*.

let *?m = "LEAST x. x $\in$ (a ‘ {n+1..})"*
let *?j = "SOME j. j > n $\wedge$ a j = ?m"*

Note the use of the indefinite description operator *SOME* in the definition of *?j*. This is justified by the fact that *?m* is a member of *a ‘ {n+1..}*. The proof continues by case analysis on *a ?j $\leq$ 9*. If that holds, then *Claim3* follows by *Claim3_a*. Otherwise we apply *Claim2*. To use it we must establish that *a ?j mod 3 $\neq$ 2*. This follows from the first sentence in the informal proof: ”First we notice that, by the definition of the sequence, a multiple of 3 is always followed by another multiple of 3”. We formalize this by the following statement:

have *"$\forall$ n n’. a n mod 3 = 0 $\wedge$ n $\leq$ n’ $\longrightarrow$ a n’ mod 3 = 0"*

Now we use induction on *n’ - n* in the proof. This proof also requires proving and using the following simple number-theoretic lemma.

lemma
  fixes *x :: nat*
  shows *"(x * x) mod 3 = 0 $\longleftrightarrow$ x mod 3 = 0"*
  ⟨proof⟩

Since *a n mod 3 = 0*, it must hold that *a ?j mod 3 = 0 $\neq$ 2*, so *Claim2* can be used to obtain *m > ?j* such that *a m < a ?j*, but this clearly contradicts the definition of *?j*.

Finally, the informal proof gives the last claim.

Claim 4. ”If $a_n \equiv 1 \pmod 3$, then there is an index $m > n$ such that $a_m \equiv -1 \pmod 3$.
Proof. In the sequence, 4 is always followed by $2 \equiv -1 \pmod 3$, so the claim is true for $a_n = 4$. If $a_n = 7$, the next terms will be 10, 13, 16, 4, 2, ... and the claim is also true. For $a_n \geq 10$, we again take an index $j > n$ such that a_j is equal to the minimum value of the set $\{a_{n+1}, a_{n+2}, \ldots\}$, which by the definition of the sequence consists of non-multiples of 3. Suppose $a_j = 1 \pmod 3$. Then we must have $a_j \leq 9$ by *Claim2* and the minimality of a_j. It follows that $a_j \in \{4, 7\}$, so $a_m = 2 < a_j$ for some $m > j$, contradicting the minimality of a_j. Therefore, we must have $a_j \equiv -1 \pmod 3$.”

Similar as we did for the *Claim3*, first we prove the following sub-claim.

have *Claim4_a: "$\forall$ n. a n mod 3 = 1 $\wedge$ a n $\leq$ 9 $\longrightarrow$ ($\exists$ m > n. a m mod 3 = 2)"*

The proof is again using direct calculations, the recursive definition of the sequence *a* and earlier established fact that *∀ n. a n > 1.*

Claim4 is then formulated as follows.

```
have Claim4: "∀ n. a n mod 3 = 1 ⟶ (∃ m > n. a m mod 3 = 2)"
```

The formal proof of this claim is very similar to the formalization of *Claim3*, so we omit the details. Formalizing the sentence ”the set $\{a_{n+1}, a_{n+2}, \ldots\}$, which by the definition of the sequence consists of non-multiples of 3”, required proving the following fact (by induction on *n' - n*).

```
have "∀ n n'. a n mod 3 ≠ 0 ∧ n ≤ n' ⟶ a n' mod 3 ≠ 0"
```

Once all four claims are formally proved, the theorem itself can be proved. Informal proof goes as follows.

Main theorem. ”It follows from the previous claims that if a_0 is a multiple of 3 the sequence will eventually reach the periodic pattern $3, 6, 9, 3, 6, 9, \ldots$; if $a_0 \equiv -1 \pmod 3$ the sequence will be strictly increasing; and if $a_0 \equiv 1 \pmod 3$ the sequence will eventually be strictly increasing. So the sequence will eventually be periodic if and only if, a_0 is a multiple of 3.”

We show the full formal proof of this part (note that *Claim3* is used in the first direction, while the second direction uses only *Claim1* and *Claim4*, while *Claim2* is only a lemma used with the proof of *Claim3* and *Claim4*, and not in the main proof).

```
show ?thesis
proof
  assume "a 0 mod 3 = 0"
  then have "eventually_periodic a" using Claim3 two_same_periodic[OF assms(1)] by simp
  then show "∃ A. infinite {n. a n = A}" using infinite_periodic[OF assms(1)] by simp
next
  assume "∃ A. infinite {n. a n = A}"
  then have "eventually_periodic a" using infinite_periodic[OF assms(1)] by simp
  {
    assume "a 0 mod 3 = 1"
    then obtain m where "a m mod 3 = 2" using Claim4 by auto
    then have False using Claim1 ‹eventually_periodic a› by force
  }
  moreover
  {
    assume "a 0 mod 3 = 2"
    then have False using Claim1 ‹eventually_periodic a› by force
  }
  ultimately show "a 0 mod 3 = 0" by presburger
qed
```

4 Educational aspects

The repository of formalized IMO problems is created with an ITP course at Faculty of Mathematics, University of Belgrade. It is an elective course at the fourth, final year of undergraduate studies. All students enrolled already passed courses in mathematical logic, functional programming, and many other classic mathematical topics during their previous studies (algebra, analysis, combinatorics, numerical mathematics etc.), so they already have quite a good understanding of all main concepts underlying ITP. The course is taught 14 weeks with 2 hours of lecture and 3 hours of exercises per week, and covers formalization of mathematics and elements of software verification in Isabelle/HOL. Grading is done by several on-site tests with small and simple problems, and a larger project, done off-site (at home), scoring up to 40% of the total points. Formalizing an IMO problem or a similar problem given at Serbian national level competitions is given as one possible project assignment (students can also verify some elementary algorithm or formalize several theorems from some introductory mathematics course).

Each student selects a problem and formalizes it. No collaboration between students is allowed, but when students get stuck, they can get help from the teachers “free of charge” (they do not lose any points for asking help). The course is new and has been given only twice, but our experience shows that most students manage to finish their project (some on their own, and some after several rounds of guidance by the teachers). Of course, it is very important that students are given informal proofs in advance, since IMO problems, although very elementary, are very challenging and hard to solve. Developed formal proofs are usually not the most elegant ones, as students often introduce definitions and lemmas that already exist in the vast Isabelle/HOL libraries and often give long, manual Isar proofs for statements that could be proved automatically if advanced automated proof methods are setup correctly. They are not penalized for this, but are sometimes required to ”polish” their proofs by following detailed guidelines given by the teachers.

Formalizing IMO problem solutions is a very good task for practicing interactive theorem proving and we advocate that they should be used in courses of formal theorem proving.

- Although they are very hard, problems are usually formulated in elementary terms of high-school mathematics and do not require any knowledge of advanced mathematical concepts. Therefore, all students of mathematics and computer science can easily understand them. Official IMO solutions do not use any advanced theorems and proof steps are justified by using elementary statements that are already available in most proof assistants.

- Formalizing a problem solution usually requires several hours. Isar proofs are usually around several hundred lines of code (the shortest proof we formalized was 95 LOC, while the longest was 2024 LOC, although that depends on the code indenting style). Therefore, such problems should not be used in limited-time on-site exams, but they are perfect for homework and project assignments.

- A rich repository of manually formalized solutions might offer a good ground for developing and training methods for automated solving of IMO problems (and hopefully contribute to the IMO grand challenge).

Although IMO competitors are high-school pupils, we do not yet have any experience in formalizing IMO solutions with that population. We suppose that teaching use of proof assistants would be too demanding. On the other hand, we think that analyzing existing problem formalizations could help the most advanced competitors in recognizing the subtlest proof details and mastering the highest level of mathematical precision and rigor, that could help reaching maximal scores in competitions.

5 Related Work

Relationship between informal proofs and their formal counterparts is often discussed in the literature. One example, is Dana Scott's Foreword of the Freek Wiedijk's seminal paper comparing 17 theorem provers [8]. Scott showed examples of proofs done by changing problem representation (e.g., algebraic and geometric) and showed examples of proofs that involve augmenting the original problem with supporting elements (e.g., auxiliary points and lines introduced in a geometric configuration), that make the original problem significantly easier to solve. Scott argued that although these changes are often easily realized and understood by humans, computers on the other hand have much difficulty in finding such proofs. A popular informal proof method is given by the so-called ”Proofs Without Words”, where the property is intuitively described by a convenient figure (e.g., there are many such diagrams that illustrate the Pythagorean theorem). One famous problem with such a proof is the calisson puzzle[6]. Although ”the proof” is very intuitive (and indeed remarkable), E. W. Dijkstra criticized that it is an example of ”an elaborate nonproof” (since key arguments are not formally stated nor proved, a slight change of the problem could yield ”the proof” incorrect).

Although the body of formalized mathematics is growing every day, examples of formalized IMO solutions or similar types of problems are scarce: Manuel Eberl has formalized three out of six problems from IMO 2019 (Q1, Q4, and Q5) [2], and several problems statements have been formally encoded in Lean (`http://github.com/IMO-grand-challenge/formal-encoding`).

One library of formalized solutions of high-school problems is presented by Pham, Bertot and Narbox [6]. They developed a dynamic geometry proving tool for interactive proving for high-school students and used a specific axiomatic system adapted to this task using the notion of vectors. Sana Stojanović-Ðurđević (the second author of the current paper), proposed a method for proving high-school problems in geometry by using coherent logic and a set of automated theorem provers to fill-in the gaps in the manually generated proof outlines [7]. They used a semi-formal TPTP-like language. The method has been successfully applied to a collection of geometric problems from Serbian high-school textbooks in geometry. Generated proofs are automatically translated to Coq and Isabelle/HOL and formally verified.

6 Conclusions and Further Work

In this paper we have presented Isabelle/HOL formalization of several official IMO problem solutions. The formalization is created within the Interactive Theorem Proving course on Faculty of Mathematics, University of Belgrade, and is available in a repository `http://github.com/filipmaric/IMO`. Our experience shows that most final year undergraduate students at the end of the course can successfully cope with such assignments, if they are given enough time, guidance and support by the teachers.

Our experience shows that the difference between formal and informal proof significantly varies, mainly depending on the category of the problem. Problems in algebra usually have very rigorous informal proofs, that are easy to formalize. Unlike those, problems in combinatorics usually give a very rough proof outline, that requires significant effort to formalize. In many cases a significant effort is required even to give a precise problem statement (problems in combinatorics usually require many introductory definitions). The proofs in number theory are somewhere in between (depending on the problem).

[6] `http://nau.edu/wp-content/uploads/sites/145/NAU-High-School-Math-Day-The-Calissons-Problem-fall-19.pdf`

If human competitors were to generate formal proofs, the competition would be much harder for them (and definitely unsuitable for high-school pupils). Our analysis shows that many official solutions (that would certainly be graded by maximal scores) are quite far from fully formal and often resort to obviousness and intuition. Therefore, in our opinion, the current formulation of the IMO grand challenge is extremely unfair for the artificial intelligence. We suggest to split the challenge in two parts:

- Informal challenge, that would require automated provers only to generate high-level proof outlines, that can be manually judged, the same way as pupils solutions are judged.
- Formal challenge, that would require provers to generate machine checkable proofs given high-level proof outlines of various granularity (the coarsest one being only the formal problem statement).

Our repository is open and we hope that a wider community of contributors will be formed. Everyone is invited to contribute either by formalizing new IMO problem statements, or by providing alternative solutions to existing problems. We assume that many existing formal proofs could be shortened and better automated and we invite contributors to provide such proofs.

References

[1] Dušan Djukić, Vladimir Janković, Ivan Matić & Nikola Petrović (2011): *The IMO compendium*, second edition. Springer, New York, doi:10.1007/978-1-4419-9854-5.

[2] Manuel Eberl (2019): *Selected Problems from the International Mathematical Olympiad 2019*. Archive of Formal Proofs. `http://isa-afp.org/entries/IMO2019.html`, Formal proof development.

[3] Jean-David Génevaux, Julien Narboux & Pascal Schreck (2011): *Formalization of Wu's Simple Method in Coq*. In Jean-Pierre Jouannaud & Zhong Shao, editors: Certified Programs and Proofs, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 71–86, doi:10.1007/978-3-642-25379-9_8.

[4] Predrag Janičić, Julien Narboux & Pedro Quaresma (2012): *The Area Method: a Recapitulation*. Journal of Automated Reasoning 48(4), pp. 489–532, doi:10.1007/s10817-010-9209-7. Available at `http://hal.archives-ouvertes.fr/hal-00426563`.

[5] Tobias Nipkow (2020 (accessed Jun 20, 2020)): *Programming and Proving in Isabelle/HOL*. Available at `http://isabelle.in.tum.de/doc/prog-prove.pdf`.

[6] Tuan-Minh Pham, Yves Bertot & Julien Narboux (2011): *A Coq-Based Library for Interactive and Automated Theorem Proving in Plane Geometry*. In Beniamino Murgante, Osvaldo Gervasi, Andrés Iglesias, David Taniar & Bernady O. Apduhan, editors: Computational Science and Its Applications - ICCSA 2011, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 368–383, doi:10.1007/978-3-642-21898-9_32.

[7] Sana Stojanović-Đurđević (2019): *From informal to formal proofs in Euclidean geometry*. Annals of Mathematics and Artificial Intelligence 85(2), doi:10.1007/s10472-018-9597-7. Available at `http://doi.org/10.1007/s10472-018-9597-7`.

[8] Freek Wiedijk (2006): *The Seventeen Provers of the World: Foreword by Dana S. Scott (Lecture Notes in Computer Science / Lecture Notes in Artificial Intelligence)*. Springer-Verlag, Berlin, Heidelberg, doi:10.1007/s11225-007-9093-2.