Лицей «Физико-техническая школа»
Санкт-Петербургского Академического университета

Курсовая работа (отчет по практике)

Формализация доказательства теоремы Зигмонди на языке Lean

Работу выполнил:
Федоров Василий (2023А)
Научный руководитель:
Васильев Артём Тарасович
Место прохождения практики:
Университет ИТМО

Санкт-Петербург, 2022

Аннотация

В ходе работы было формализовано доказательство теоретико-числовой теоремы Зигмонди, утверждающей, что почти для всех пар натуральных n и a найдётся простое p такое, что a имеет показатель n по модулю p.

Для формализации использован инструмент интерактивного доказательства теорем *Lean theorem prover*. В ходе работы также были формализованы некоторые смежные с теоремой Зигмонди леммы о многочленах деления круга.

Введение 4
- Инструменты интерактивного доказательства теорем . . . 4
- Lean . . . 4
- mathlib . . . 6

Постановка задачи 7
- Определения . . . 7
- Теорема Зигмонди . . . 7
- Формулировка на Lean . . . 7

Формализация доказательства 9

Результат работы 12

Благодарности 13

Список литературы 14

Введение

Инструменты интерактивного доказательства теорем

Инструменты интерактивного доказательства теорем (*proof assistants*, далее PA) - компьютерные программы, позволяющие пользователю формально записывать необходимые определения, формулировать теоремы и доказательства. Затем программа проверяет логические переходы и действительно ли они ведут к необходимому утверждению из уже доказанных фактов [**2**].

Обычно написание формального доказательства в PA - значительно более трудоёмкий процесс, чем изложение тех же рассуждений на бумаге, поскольку программе надо "объяснять"многие детали, которые человеку кажутся очевидными и обычно опускаются. С другой стороны, в современной математике нередко встречаются такие длинные доказательства, которые человеку (а для верификации проверить должен не один человек) проверить оказывается трудно - особенно если используется длинный компьютерный перебор (например, в доказательстве знаменитой Теоремы о четырёх красках для проверки перебора случаев используется система *Coq* [**3**]). Более того, бывает и так, что неверные утверждения проходят проверку и публикуются в крупных журналах (в Annals of Mathematics, например, опубликована статья *Non-quasi-projective moduli spaces*, прямо в аннотации которой указана более ранняя статья, которой та противоречит). Со временем математические доказательства становятся сложнее, могут чаще включать использование компьютерных программ, из-за чего развитие PA становится всё более актуальным.

На данный момент существует более десятка PA. В работе был использован язык под названием *Lean*.

Lean

Lean (полное название - *Lean theorem prover*) - PA, разработанный Леонардо де Моурой в Microsoft Research на языке C++ в 2013 году [**1**]. Lean основан на исчислении индуктивных конструкций, то есть оперирует выражениями, состоящими из некоторых *термов* (утверждения, переменные), каждая из которых имеет свой *тип*, который может быть индуктивным (то есть определён через начальные условия и правило получения новой термы из предыдущих). Например, натуральные числа в Lean определены следующим образом:

```
inductive nat : Type
| zero : nat
| succ : nat → nat
```

Здесь слово `inductive` означает, что объявленный тип `nat` (от *natural*) является индуктивным. Затем определяется первое натуральное число - `zero` (в Lean число 0 натуральное) и терма `succ` (от *successor*), которая по натуральному числу (элементу `nat`) выдаёт следующий элемент (то есть принимает на вход число n и возвращает $n+1$). Далее в библиотеке языка `mathlib` определяются сложение, умножение, неравенства натуральных чисел и т. д.

Кроме определений в Lean есть и сами доказательства. Рассмотрим их на примере инъективности умножения в натуральных числах, то есть если $a \cdot b = a \cdot c$ и $a \neq 0$, то $b = c$.

```
lemma mul_left_cancel (a b c : ℕ) (ha : a ≠ 0) : a * b = a * c → b = c :=
begin
  induction c with d hd generalizing b,
  {
    simp [or_iff_right ha],
  },
  {
    cases b,
    {
      simp,
      intro ha',
      exfalso,
      exact ha ha',
    },
    {
      repeat { rw nat.succ_eq_add_one, },
      repeat { rw [mul_add, add_right_cancel_iff], },
      exact hd b,
    }
  }
end
```

Доказательства в Lean состоят из последовательного применения *тактик*. Первая применённая тактика - `induction c`, она позволяет перейти к доказательству утверждения индукцией по c. Ей передаются аргументы - как будет названа переменная, по которой идёт индукция и как обращаться к индукционному предположения. Ещё один аргумент - `generalizing b` означает, что вместо доказательства для конкретного b на каждом шаге индукции утверждение выводится для всех b сразу, и в итоге индукционным предположением будет утверждение вида «для любого b...». После применения тактики `induction` появляется вместо изначальной цели две (база и переход), поэтому далее открываются фигурные скобки для каждой цели.

В доказательстве базы используется тактика `simp` (от английского *simplify*). При её применении Lean ищет леммы, которые могут упростить текущую цель (такие в библиотеке отдельно помечены) и применяет их. В квадратных скобках можно указать, какие леммы следует использовать дополнительно к стандартным.

Тактика `cases` разбивает задачу на несколько случаев в зависимости от того, как сконструирована переменная индуктивного типа, которая передана в качестве аргумента. В данном случае она принимает натуральное число b и сводит лемму к двум случаям: в одном $b = 0$, в другом b - следующее после какого-то числа.

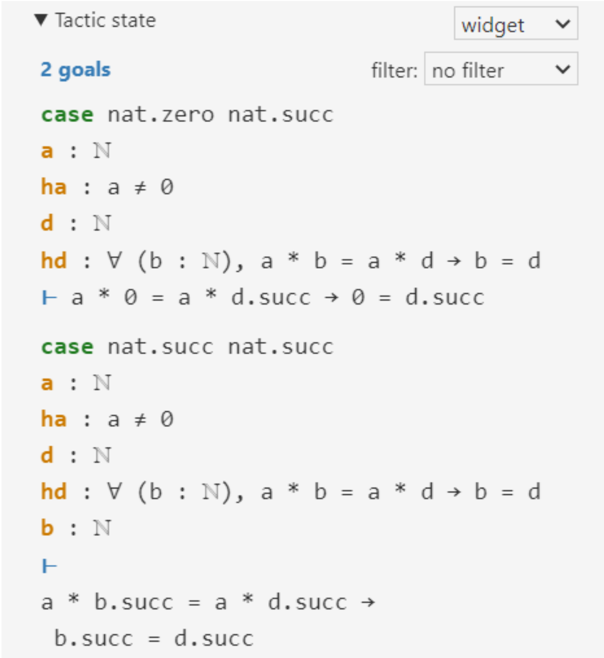

Рис. 1: Прогресс доказательства отображается в отдельном поле редактора кода *Lean infoview*, в котором перечислены локально доказанные утверждения, имеющиеся условия (*локальный контекст*) и текущая цель.

Тактики `intro`, `rw` (от *rewrite*), `exact` позволяют манипулировать утверждениями в задаче. `intro` перемещает терму, из цели в локальный контекст (например, если цель является импликацией $A \to B$, она станет B, а в локальном контексте появится A). `rw` принимает в качестве аргумента лемму вида $x = y$ или $x \leftrightarrow y$, находит в цели x и заменяет на y. `exact` позволяет завершить доказательство, если цель уже совпадает с утверждением, которое тактике передано как аргумент.

Тактика `repeat` повторяет последовательность действий в фигурных скобках столько раз, сколько возможно - например, если необходимо несколько раз применить одну и ту же лемму. `exfalso` заменяет текущую цель на `false` и применяется, когда текущий локальный контекст уже приводит к противоречию.

mathlib

Все леммы и теоремы, которые уже формализованы в Lean и могут пригодиться в следующих доказательствах, собраны в открытой библиотеке `mathlib`. Библитека, как видно по её репозиторию на *Github*, активно развивается. Значимые доказанные результаты перечислены на сайте Lean. Кроме того, `mathlib` уже использовалась в формализации доказательства ещё недавно открытых проблем: например, гипотеза Эрдёша — Грэма на тему разбиения числа 1 в суммы слагаемых вида $\frac{1}{n}$ в 2021 году была доказана не только в самой статье [4], но и с помощью Lean.

Постановка задачи

Определения

В доказательстве нам понадобятся следующие определения:

- *Показатель* или *порядок* числа a по модулю m, взаимно простому с a - наименьшее натуральное число r такое, что $m \mid a^r - 1$;
- *n-й многочлен деления круга* или *круговой многочлен* - многочлен, равный

$$\prod_{\text{НОД}(i,n)=1}^{i<n} (x - e^{\frac{2\pi i}{n}}),$$

  то есть унитарный многочлен, корнями которого являются такие комплексные корни из единицы, что при возведении в степень меньшую, чем n, не получается 1 (они же *первообразные* корни из единицы).

Теорема Зигмонди

Целью работы является формализация доказательства теоремы Зигмонди [**5**] на языке Lean. Сама теорема формулируется в двух видах, первый - менее общий, он и будет доказываться:

Для любых натуральных чисел a, $n > 1$ существует простое p такое, что n является показателем a по модулю p, за исключением следующих ситуаций:

- $n = 2,\ a = 2^s - 1$, где $s \geq 2$
- $n = 6,\ a = 2.$

В более общем виде рассматривается делимость на p числа $a^n - b^n$ вместо $a^n - 1$ (то есть показатель вычета $\frac{a}{b}$), и эту версию можно легко вывести из первой, поскольку будет ясно, что вместо натурального a можно подставить рациональное $\frac{a}{b}$ и провести те же рассуждения.

Формулировка на Lean

На самом же Lean теорема формулируется менее прозрачно:

```
theorem exists_prime_of_order (hn : 1 < n) (ha : 1 < a)
(h_exception_1 : ¬(n = 2 ∧ (∃ (s : ℕ), a = 2 ^ s - 1)))
(h_exception_2 : ¬(n = 6 ∧ a = 2)) : ∃ (p : ℕ) (h_coprime : (a.coprime p)),
(nat.prime p) ∧ order_of(zmod.unit_of_coprime a h_coprime) = n
```

Начинается формулировка с ключевого слова `theorem`, по которому Lean поймёт, что далее следует теорема. Затем пишется её название (в Lean принято называть теоремы не именами их авторов, а согласно общей конвенции, чтобы их было проще искать в библиотеке). Затем задаются условия теоремы:

- a и n - переменные, обозначающие натуральные числа;
- далее идут факты, которые надо передать теореме, чтобы ей воспользоваться: $1 < n$, $1 < a$ и упомянутые выше исключения;
- наконец, после двоеточия идёт само утверждение: существование числа p, которое является простым и показатель a по модулю которого равен n.

В ходе доказательства теоремы также необходимо было сформулировать и доказать промежуточные леммы, отсутствовавшие в `mathlib`.

Формализация доказательства

Далее пройдём по плану использованного для формализации доказательства теоремы Зигмонди.

Основная идея состоит в следующей равносильности:

Если простое p не делит n, то показатель a по модулю p равен n тогда и только тогда, когда $p \mid \Phi_n(a)$, где $\Phi_n(x)$ - n-й многочлен деления круга.

Хоть этот факт и является одним из начальных в теории круговых многочленов, в момент начала работы в `mathlib` его не было и пришлось доказывать его в ходе практики. В итоге формализация недостающих теорем о многочленах деления круга, включая эту, заняло около трети всего объёма проделанной работы.

Ясно, что теперь, пойдя от противного, нам достаточно привести к противоречию предположение о том, что любой простой делитель $\Phi_n(a)$ делит ещё и n. Теперь докажем из этого, что такое простое всего лишь одно. Для этого воспользуемся ещё одним фактом про многочлены деления круга: если $p \mid \Phi_n(a)$, p простое и r - показатель a по модулю n, то n представимо в виде $n = rp^t$.

```
have h_n_eq_ord_mul_pow : ∀ (p' : ℕ) (h_in_factors : p' ∈ Φ.factors),
 ∃ (t : ℕ), n = order_of (zmod.unit_of_coprime a
(h_coprime p' (nat.dvd_of_mem_factors h_in_factors))) * p' ^ t ∧ 1 ≤ t,
{
  intros p' h_in_factors,
  set a_unit := zmod.unit_of_coprime a
  (h_coprime p' (nat.dvd_of_mem_factors h_in_factors)) with h_a_def,
  have h_root : (polynomial.cyclotomic n (zmod p')).is_root a_unit,
  {
    simp only [zmod.coe_unit_of_coprime, polynomial.is_root.def],
    rw [← polynomial.map_cyclotomic_int, polynomial.eval_nat_cast_map,
    eq_int_cast, zmod.int_coe_zmod_eq_zero_iff_dvd,
    ← int.to_nat_of_nonneg (polynomial.cyclotomic_nonneg n h_one_le_a_int),
    int.coe_nat_dvd, ← h_Phi_def],
    exact nat.dvd_of_mem_factors h_in_factors,
  },
  have h_p'_prime_fact : fact (p'.prime) :=
  by { rw fact_iff, exact nat.prime_of_mem_factors h_in_factors, },
  cases prime_dvd_cyclotomic hpos_n h_p'_prime_fact h_root with t ht,
  use t,
  split,
  { exact ht, },
    {
      by_contra,
      simp only [not_le, nat.lt_one_iff] at h,
      subst h,
      simp only [pow_zero, mul_one] at ht,
      haveI : ne_zero p' := ⟨ nat.prime.ne_zero (nat.prime_of_mem_factors h_in_factors) ⟩,
      have h_order_lt_p' : order_of (zmod.unit_of_coprime a
      (h_coprime p' (nat.dvd_of_mem_factors h_in_factors))) < p',
      {
        have h_p'_sub_one_lt : p' - 1 < p' := by exact nat.sub_lt
        (nat.prime.pos (nat.prime_of_mem_factors h_in_factors)) zero_lt_one,
        have h_order_le_p'_sub_one :
        order_of (zmod.unit_of_coprime a (h_coprime p'
        (nat.dvd_of_mem_factors h_in_factors))) ≤ p' - 1 := by {
          rw ← nat.totient_prime (nat.prime_of_mem_factors h_in_factors),
          exact order_of_units_le_totient (h_coprime p'
          (nat.dvd_of_mem_factors h_in_factors)),
        },
        exact lt_of_le_of_lt h_order_le_p'_sub_one h_p'_sub_one_lt,
      },
      apply nat.not_dvd_of_pos_of_lt
      (order_of_units_pos (h_coprime p' (nat.dvd_of_mem_factors h_in_factors))) h_order_lt_p',
      rw ← ht,
      exact h_p_dvd p' h_in_factors,
    }
  },
```

Рис. 2: Фрагмент доказательства теоремы Зигмонди, в котором выводится, что для любого простого p', делящего $\Phi_n(a)$, n представится как rp'^t

Если же у $\Phi_n(a)$ есть простые делители p и q, то получаем $n = r_1 p^{t_1} = r_2 q^{t_2}$. Не умаляя общности, пусть $p < q$. Но поскольку r_1 - показатель по модулю p некоторого числа, он точно меньше p, а значит и меньше q. Но тогда q не делит $r_1 p^{t_1} = n$, противоречие.

Таким образом, $\Phi_n(a)$ является степенью некоторого простого p. Следующим шагом является доказательство того, что $\Phi_n(a)$ не может делиться на p^2. Для этого запишем очевидную из определения многочлена деления круга делимость:

$$\Phi_n(a) \mid \frac{a^n - 1}{a^{n/p} - 1}.$$

Применив лемму об уточнении показателя, при $p \neq 2$ получаем необходимое, а случай $p = 2$ сводится к первому исключению. Если $\Phi_n(a)$ не делится на p^2 и имеет лишь один простой делитель, то, очевидно, $\Phi_n(a) = p$. Остаётся только оценить значение $\Phi_n(a)$ снизу и получить противоречие (или свести ко второму исключению).

До написания этой работы над теоремой Зигмонди уже начиналась работа, но до конца доведена не была. Поэтому в библиотеке уже имелись доказанные оценки на значение кругового многочлена:

$$(a-1)^{\phi(n)} \leq \Phi_n(a) \leq (a+1)^{\phi(n)},$$

которые легко получить, разложив многочлен деления круга на множители из $\mathbb{C}[x]$. Переписав $\Phi_n(a)$ как

$$\Phi_n(a) = \Phi_{rp^t}(a) = \frac{\Phi_r(a^{p^t})}{\Phi_r(a^{p^{t-1}})} \geq \left(\frac{a^{p^t} + 1}{a^{p^{t-1}} - 1}\right)^{\phi(n)},$$

получаем неравенство

$$\left(\frac{a^{p^t} + 1}{a^{p^{t-1}} - 1}\right)^{\phi(n)} \leq p.$$

Опустим технические подробности получения противоречия с этого места.

```
lemma three_mul_le_pow_sub (x : ℕ) : (2 + 1) * ((x + 5) : ℤ) ≤ 2 ^ (x + 5) - 1 :=
begin
  induction x with x hi,
  {
    norm_num,
  },
  {
    transitivity (2 ^ (x + 5) - 1 + ((2 : ℤ) + 1)),
    {
      rw [nat.cast_succ, add_assoc],
      nth_rewrite 2 add_comm,
      rw [← add_assoc, mul_add, mul_one],
      exact int.add_le_add_right hi _,
    },
    {
      rw [sub_add_add_cancel, nat.succ_eq_add_one],
      nth_rewrite 3 add_comm,
      rw [add_assoc],
      repeat { rw pow_add, },
      set two_pow := (2 : ℤ) ^ x with h_pow,
      have h_pow_nonneg: 1 ≤ two_pow,
      {
        rw [h_pow, ← nat.cast_one, ← nat.cast_bit0, ← nat.cast_pow,
        nat.cast_le],
        apply nat.one_le_pow,
        linarith,
      },
      norm_num,
      linarith,
    }
  }
end
```

Рис. 3: Доказательство неравенства $3(x+5) \leq 2^{x+5} - 1$ при натуральных x. На первый взгляд кажется очевидным, но формальное доказательство по индукции занимает некоторое время.

Результат работы

В итоге удалось формализовать приведённое выше доказательство (см. репозиторий на *Github*). Все понадобившиеся по пути леммы тоже доказаны.

Кроме самого доказательства планировалось внести вклад в развитие `mathlib`, запросив добавление написанного кода в библиотеку. К сожалению, сделать это не удалось, потому что незадолго до окончания работы выяснилось, что ещё один, более опытный энтузиаст Lean завершает формализацию той же теоремы.

Благодарности

Выражаю благодарность своему научному руководителю Артёму Тарасовичу Васильеву за обучение программированию на Lean, помощь в решении возникающих во время доказательства проблем и в целом активное участие в практике.

Список литературы

[1] Lean prover community (веб-ресурс, URL: https://leanprover-community.github.io, дата обращения: 20.12.2022)

[2] *Herman Geuvers* - Proof assistants: History, ideas and future. *Sādhanā* Vol. 34, Part 1, February 2009

[3] *Georges Gonthier* - A computer-checked proof of the Four Colour Theorem (2005), Microsoft Research Cambridge

[4] *Thomas F. Bloom* - On a density conjecture about unit fractions (2021), Mathematical Institute, Woodstock Road, Oxford

[5] *Karl Zsigmondy* - Zur Theorie der Potenzreste. Monatshefte für Mathematik und Physik Vol. 3, 265–284 (1892)