Математическая логика и теория алгоритмов

Осень 2016, лектор: Д.М.Ицыксон

Авторы: Егор Суворов, Надежда Бугакова, Елизавета Третьякова, Юрий Кравченко, Анастасия Старкова

Собрано: 11 февраля 2016 г. 12:04

Оглавление

0.1. Вступление

В этом семестре курс будет состоять из трёх частей:

1. Вычислимость (теория алгоритмов)
2. Вычисление предикатов
3. Теория множеств

Рекомендуется для прочтения: Н. К. Верещагин, А. Шень. "Лекции по математической логике и теории алгоритмов". Читать в следующем порядка:

1. Часть 3. Вычислимые функции.

   `http://www.mccme.ru/free-books/shen/shen-logic-part3-2.pdf`

2. Часть 2. Языки и исчисления.

   `http://www.mccme.ru/free-books/shen/shen-logic-part2-2.pdf`

3. Часть 1. Начала теории множеств

   `http://www.mccme.ru/free-books/shen/shen-logic-part1-2.pdf`

Это не ошибка: от третьей части к первой. Потому что гладиолус :)

Глава 1

Вычислимость

1.1. Введение

Будем пользоваться интуитивным определением понятия *Алгоритм*. Скажем лишь то, что он удовлетворяет некоторым условиям:

1. Получает на вход строку из Σ^* — множества строк над заданным алфавитом Σ
2. Либо закончит работу и выдаст строку из Σ^*, либо никогда не закончит работу
3. Его можно записать
4. Его можно исполнять по шагам
5. Может использовать бесконечное количество памяти

𝔇𝔢𝔣 1.1.1. Обозначим множество целых неотрицательных чисел:

$$\mathcal{N} = \mathbb{N} \cup \{0\}$$

Счётных множеств у нас будет много. Например, построим биекцию между строками и натуральными числами на примере $\Sigma = \{a, b\}$:

- $0 \to$ пустая строка
- $1 \to a$
- $2 \to b$
- $3 \to aa$
- $4 \to ab$
- ... и так далее

Дальше будем считать, что алгоритмы могут вместо строк принимать на вход элементы $\mathcal{N}$ — как нам удобнее.

1.2. Разрешимые, перечислимые множества. Вычислимые функции

Def 1.2.1. $A \subset \mathcal{N}$ называется *разрешимым*, если существует алгоритм, который выдаёт единицу для чисел, лежащих в A, и ноль для чисел, не лежащих в A:

$$\exists Algo\colon Algo(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

Замечание 1.2.1. Время работы алгоритма нам сейчас неинтересно: лишь бы оно было конечно.

Примеры разрешимых множеств:

1. Пустое множество: алгоритм, выводящий 0
2. $\mathbb{N}$: алгоритм, выводящий 1
3. Произвольное конечное множество A: алгоритм, по очереди сравнивающий параметр с каждым элементом
4. $\{q \in \mathbb{Q} \mid q < e\}$ (как мы помним, рациональных чисел тоже счётно, поэтому можно считать, что алгоритму на вход даётся некоторое рациональное число)

   ▶ В самом деле, мы умеем сколь угодно хорошо приближать e следующими соотношениями:
   $$\left(1 + \frac{1}{n}\right)^n < e < \left(1 + \frac{1}{n}\right)^{n+1}$$
   Значит, если у нас есть некоторое рациональное число (которое точно не равно e), то, взяв достаточно большое (конечное!) n, мы сможем определить, с какой стороны от e лежит аргумент. ◀
5. $\{n \in \mathbb{N} \mid$ в записи числа π есть n девяток подряд$\}$

   ▶ Очевидно, что либо в π встречается любое количество девяток подряд (и тогда алгоритм тривиален), либо есть минимальное N такое, что N девяток подряд нет. Тогда алгоритм просто сравнивает аргумент с этим N.
   Это пример неконструктивного доказательства: алгоритм существует, но предъявить мы его не смогли (равно как и явно запустить) — про число π такой факт неизвестен. ◀

Def 1.2.2. $A \subset \mathcal{N}$ называется *полуразрешимым*, если

$$\exists Algo\colon \begin{cases} Algo(x) = 1, & x \in A \\ Algo(x) \text{ работает бесконечно долго}, & x \notin A \end{cases}$$

Замечание 1.2.2. A разрешимо $\Rightarrow A$ полуразрешимо

Def 1.2.3. $A \subset \mathcal{N}$ называется *перечислимым* если существует алгоритм, который на входе 0 работает бесконечно долго и выводит все элементы множества (для каждого конкретного элемента есть конкретное время, в которое его выведут).

Теорема 1.2.1. Пусть $S \subset \mathbb{N}$, тогда S перечислимо $\iff$ S полуразрешимо

▶ **$\Rightarrow$:** Пусть B — перечисляющий алгоритм. Опишем полуразрешающий:

```
Пока B(0) не выдало x {
  ждём
}
Выводим 1
```

⇐: Пусть A — полуразрешающий алгоритм. Опишем перечисляющий:

```
Для i = 0..INF {
  Для j = 0..i {
    запустить первые i - j шагов A(j)
    если завершился, то выводим j
  }
}
```

Это мы просто пошли по диагоналям таблицы «номер входа/номер шага». Интуиция: можно считать, что мы запустили полуразрешающий алгоритм на всех входах одновременно.

◀

Теорема 1.2.2 (Поста)**.** Всякое разрешимое множество натуральных чисел перечислимо. Если множество A и его дополнение (до множества всех натуральных чисел) перечислимы, то A разрешимо.

▶ 1. Если принадлежность числа к множеству A можно проверить некоторым алгоритмом, то A и его дополнение перечислимы: надо по очереди проверять принадлежность чисел 0, 1, 2, ... и печатать те из них, которые принадлежат A (или те, которые не принадлежат A).

2. Если у нас есть алгоритм, перечисляющий A, а также другой алгоритм, перечисляющий дополнение к A, то для выяснения принадлежности заданного числа n к A надо запустить оба эти алгоритма параллельно (исполнять шаг одного, потом шаг другого) и ждать, пока один из них напечатает n (мы знаем, что рано или поздно ровно один из них это сделает). Посмотрев, какой алгоритм это сделал, мы узнаем, лежит ли n в A. Это произойдёт за конечное время.

◀

Теорема 1.2.3. Множество P натуральных чисел перечислимо тогда и только тогда, когда оно является проекцией некоторого разрешимого множества Q пар натуральных чисел. Проекция получается, если от пар оставить их первые компоненты:

$$P \text{ перечислимо}$$
$$\Updownarrow$$
$$\exists \text{ разрешимое } Q \subset \mathcal{N} \times \mathcal{N} \colon x \in P \iff (\exists y \colon (x, y) \in Q)$$

▶ ⇑: Проекция любого перечислимого множества перечислима (перечисляющиий алгоритм должен лишь удалять вторые члены пар), так что проекция разрешимого множества тем более перечислима.

⇓: Если P — перечислимое множество, перечисляемое алгоритмом A, то оно есть проекция множества Q, состоящего из всех таких пар (x, n), что x появляется в течении первых n шагов работы алгоритма A. Это множество, очевидно, разрешимо.

◀

Def 1.2.4. Пусть $M \subset \mathcal{N}$. Тогда функция $f \colon M \to \mathcal{N}$ называется *вычислимой*, если существует алгоритм, ее вычисляющии, то есть такой алгоритм A, что:

1. если $n \in M$, то алгоритм A останавливается на входе n и печатает $f(n)$

2. если $n \notin M$ (функция не определена), то алгоритм A не останавливается на входе n

Несколько простых свойств в качестве упражнений:

1. S перечислимо $\iff$ S — область определения вычислимой функции f.

   ▶ **⇒:** S перечислимо, следовательно полуразрешимо. Возьмём функцию f, соответствующую полуразрешимому алгоритму — она определена в точности на S.

   ⇐: Возьмём алгоритм, вычисляющий функцию f и заменим его вывод на единицу — получим полуразрешающий алгоритм для области определения f.

   ◀

2. S перечислимо $\iff$ S — область значений вычислимой функции f

   ▶ **⇒:** Модифицируем полуразрешающий алгоритм для S: пусть вместо единицы он выводит x.

   ⇐: По предыдущему свойству область определения f перечислима. Запустим перечисляющий алгоритм для области определения, на каждом его выводе дополнительно вычислим f (за конечное время, так как функция определена). Получим алгоритм, выводящий область значений.

   ◀

Теорема 1.2.4. Пусть $f\colon M \to \mathcal{N}$ вычислима $\iff$ $\Gamma_f = \{(x, f(x)) \mid x \in M\}$ (график функции) перечислим.

▶ **⇒:** Так как f вычислима, то область определения вычислима, перечислим её. Как только алгоритм выводит x, вычисляем $f(x)$ и выводим $(x, f(x))$.

⇐: Чтобы вычислить значение $f(x)$ запустим алгоритм, перечисляющий точки графика. За конечное время он выведет пару (x, y) (с необходимым x), тогда выводим y, как ответ.

◀

1.3. Нумерация алгоритмов

Для удобства хочется получить некоторое соответствие между алгоритмами и элементами $\mathcal{N}$. Довольно естественно считать, что алгоритм можно задать некоторой строкой (например, конечной программой). Строгое определение мы дадим позже. Строки, не являющиеся корректными описаниями, будем считать соответствующими пустому алгоритму, который сразу завершается и ничего не выводит. Так как все строки мы можем занумеровать элементами $\mathcal{N}$, то и все возможные алгоритмы мы тоже можем занумеровать.

𝔇𝔢𝔣 1.3.1. Пусть A — алгоритм. Тогда за $\#A$ мы обозначим *номер алгоритма*, некоторый элемент $\mathcal{N}$, ему соответствующий.

𝔇𝔢𝔣 1.3.2. Пусть n — натуральное число. Тогда *алгоритм с номером* n мы будем обозначать $\langle n \rangle$.

𝔇𝔢𝔣 1.3.3. *Универсальный алгоритм* $U(n, x) = \langle n \rangle(x)$; запускает алгоритм с номером n на входе x.

𝔇𝔢𝔣 1.3.4. *Диагональная функция* $u(n) = U(n, n)$; скармливает алгоритму с номером n на вход его собственный номер.

Замечание 1.3.1. $u(n)$ определена не везде, например, есть алгоритмы, которые никогда не останавливаются.

Теорема 1.3.1. Функцию u нельзя вычислимо доопределить до всюду определённой функции.

▶ Пусть $u\colon W \to \mathcal{N}$, а мы хотим найти такую $u'\colon \mathcal{N} \to \mathcal{N}$, что $\forall x \in W\colon u'(x) = u(x)$.

Доказательство от противного: пусть есть вычислимая u'. Давайте рассмотрим функцию $d(n) = u'(n) + 1$ — она также вычислима и во всех точках отличается от u. Пусть $\#d$ — номер алгоритма, вычисляющего d. Так как d вычислима, то $\#d \in W$. Тогда рассмотрим u в точке $\#d$:

$$u(\#d) = \langle \#d \rangle(\#d) = d(\#d) = u'(\#d) + 1 \neq u(\#d)$$

Получаем противоречие, так как в точке $\#d$ значения u и u' должны совпадать. ◀

Следствие 1.3.1.1. Множество $W = \{n \mid \langle n \rangle(n)$ останавливается$\}$ перечислимо, но не разрешимо.

▶ В самом деле, это множество — просто область определения диагональной функции, которая является вычислимой. Если бы W было разрешимым, то можно было бы доопределить u до всюду определённой (взяв разрешающий алгоритм для W и скомбинировав его с u), что противоречит только что доказанной теореме. ◀

Следствие 1.3.1.2. Множество $\overline{W} = \{n \mid \langle n \rangle(n)$ не останавливается$\}$ неперечислимо.

▶
- Если бы $\overline{W}$ было бы перечислимо и W было бы перечислимо, то W было бы разрешимо, что неверно.
- Альтернативное доказательство: пусть $\overline{W}$ перечислимо $\Rightarrow$ полуразрешимо. Пусть A — полуразрешающий алгоритм. Тогда возможны два варианта:

$$\left[\begin{array}{ll} \#A \in W \Rightarrow A(\#A) & \text{останавливается} \Rightarrow \#A \in \overline{W}\text{, противоречие} \\ \#A \in \overline{W} \Rightarrow A(\#A) & \text{не останавливается} \Rightarrow \#A \in W\text{, противоречие} \end{array}\right.$$

◀

1.4. m-сведение

Теперь мы хотим научиться доказывать неразрешимость других множеств. Делать это будем при помощи метода сведения: сводим неразрешимую задачу к искомой и тем самым доказываем, что искомая неразрешима.

Def 1.4.1. Пусть $A, B \subset \mathcal{N}$, тогда A *m-сводится к* B (обозначается $A \leqslant_m B$), если существует вычислимая и всюда определённая функция f такая, что:

$$x \in A \iff f(x) \in B$$

Замечание 1.4.1. Интуиция за определением такая: B в некотором смысле «не проще» A, т.е. решили $B \Rightarrow$ решили A.

Свойства сведения:

1. Если $A \leqslant_m B$ и B разрешимо, то A разрешимо.

   ▶ Раз B разрешимо, то есть разрешающий алгоритм B'. Построим разрешающий алгоритм A': $A'(x) = B'(f(x))$ — всегда завершается. ◀

2. Если $A \leqslant_m B$ и B перечислимо, то A перечислимо.

   ▶ Раз B перечислимо, то есть полуразрешающий алгоритм B'. Построим полуразрешающий алгоритм A' в точности как в предыдущем пункте. Значит, A полуразрешимо и, следовательно, перечислимо. ◀

3. Транзитивность: если $A \leqslant_m B \leqslant_m C$, то $A \leqslant_m C$.

▶ Пусть есть функция f для m-сведения A к B и функция g для m-сведения B к C. То есть:

$$\left.\begin{array}{l} x \in A \iff f(x) \in B \\ y \in B \iff g(y) \in C \end{array}\right\} \Rightarrow x \in A \iff g(f(x)) \in C$$

◀

Пример 1.4.1. Называется «проблема останова» («halting problem»).

$$H = \{(n, x) \mid \langle n \rangle(x) \text{ останавливается}\}$$

▶ Выполним m-сведение W к H (которое про остановку алгоритма на входе из себя самого):

$$f(n) = (n, n)$$
$$n \in W \iff f(n) = (n, n) \in H$$

Мы уже знаем, что W перечислимо и неразрешимо, значит, H тоже перечислимо и неразрешимо. ◀

Замечание 1.4.2. Вообще говоря, перечислимость H можно доказать проще: имеется очевидный полуразрешающий алгоритм: для пары (n, x) запускаем $\langle n \rangle(x)$ и заменяем вывод на единицу.

Пример 1.4.2. $H_0 = \{n \mid \langle n \rangle(0) \text{ останавливается}\}$.

▶ Покажем, что $W \leqslant_m H_0$. Хотим следующего:

$$n \in W \iff \langle n \rangle(n) \text{ останавливается} \iff f(n) \in H_0 \iff \langle f(n) \rangle(0) \text{ останавливается}$$

Тогда скажем, что

$$f(n) = \#\{\text{алгоритм, независимо от входа запускающий } \langle n \rangle(n)\}$$

◀

Пример 1.4.3. Пусть $H_A = \{x \mid A(x) \text{ останавливается}\}$, где A — алгоритм. Тогда существует алгоритм A такой, что H_A неразрешимо.

▶ Возьмём $A = U(n, x)$. Сведём задачу останова ($H \leqslant_m H_a$):

$$f(n, x) = (n, x)$$
$$(n, x) \in H \iff \langle n \rangle(x) \text{ останавливается} \iff U(n, x) \text{ останавливается} \iff$$
$$\iff (n, x) \in H_a \iff f(n, x) \in H_a$$

◀

1.5. Формальное определение алгоритма

1. Программа с конечным количеством переменных, массивов нет
2. Все переменные принимают произвольные значения из $\mathcal{N}$
3. Команды: `a++`, `a--` (для конкретных переменных)
4. Строки явно нумеруются (как в старых языках вроде BASIC)
5. Есть команда `goto(`n`)` (где n — не переменная, а число), обозначает «перейти к строке с номером n»

6. Есть условный оператор `if (a > 0) then goto` s_1 `else goto` s_2, где s_1 и s_2 — константы, сравнение только с нулём и только одного типа

7. Есть команда `stop`, завершающая программу

Будем считать, что все переменные в начале забиты 0, в переменной x содержится вход, а если программа остановилась (`stop`), то в y будет выход.

Пример 1.5.1. Можно выразить все «естественные» операции из языков программирования:

1. Присваивание переменной нулю: `z := 0`.

```
1. if z > 0 then goto 2 else goto 4
2. z--
3. goto 1
```

2. `a := b`

```
10. a := 0
20. c := 0
30. while (b > 0)
40.     b--
50.     a++
60.     c++
70. while (c > 0)
80.    c--
```

3. `a := b + c`

```
1. a := b
2. d := c
3. while (d > 0)
4.    d--
5.    a++
```

4. `a := bc`

5. Возведение в степень

6. Деление с остатком

7. Проверка на простоту

8. Вычитание

9. Сравнение

10. Нахождение n-го простого числа

11. Массивы! $(\alpha_1, \dots, \alpha_k) \to p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ (где p_i — различные простые числа)

1.6. Задача FRACTRAN

Придумана Джоном Конвеем, автором игры «Жизнь». Вообще говоря, это даже язык программирования.

Имеется $k+1$ переменных: $r_1, r_2, \dots, r_k > 0 \in \mathbb{Q}$ и переменная $m \in \{1, 2, 3, \dots\} = \mathbb{N}$.

За один ход ищется такое минимальное i, что $mr_i \in \mathbb{N}$ и значение m заменяется на mr_i.

Вопрос: по начальным данным ($\{r_i\}$ и m) определить, закончиться ли процесс?

Пример 1.6.1.

1. $\frac{2}{3}\ \frac{3}{2}$, m = 2. Процесс не остановится.
2. $\frac{2}{3}\ \frac{1}{17}$, m = 3. Процесс остановится.

Утверждение 1.6.1. Нет алгоритма, решающего эту задачу.

▶ Пусть $FRACTRAN$ = множество таких $\{r_i\}$ и m, что процесс остановится. Если мы покажем, что H сводится к $FRACTRAN$, то мы победили. То есть нужно по программе (для задачи останова) построить что-то аналогичное на FRACTRAN.

Def 1.6.1. Состояние памяти программы — это значение всех переменных и номер строки программы.

Пусть у нас три переменных, они соответсвуют числу $2^a3^b5^c$. Тогда номер строки мы будем обозначать $p \in \mathbb{P}, p > 7$. То есть нумеровать строки простыми числами, начиная с 7. Для большего количества переменных — аналогично.

Тогда состояния памяти можно записать, как число $m = 2^a3^b5^cp$, где p — номер строки. Теперь закодируем команды.

- ```
  7.  a++ // текущая команда
  11.     // следующая команда
  ```
  Кодируем дробью $\frac{2\cdot11}{7}$. Если сработало, то $m = 2^a3^b5^c7 \to m = 2^{a+1}3^b5^c11$.
- ```
  7.  a--
  11.
  ```
  Кодируем $\frac{11}{2\cdot14}$.
- ```
  7. goto 13
  ```
  Кодируем $\frac{13}{7}$.
- ```
  7. if a > 0 then goto 13 else goto 17
  ```
  Рассмотрим первую ветку. По-хорошему надо проверить, что m делится на 2, и тогда перейти на нужную строчку. Хороший способ — попробовать поделить. Если число было нечётное, то после домножения на $\frac{13}{2\cdot}$ оно станет нецелым и по правилу мы не сможем применить это преобразование. Но тогда значение a уменьшится на единицу, если мы-таки пойдём по первой ветке, что не очень хорошо.
  Но мы говорим, что ничего страшного, так как можно считать, что условный оператор так и работает и переписать код под новый условный оператор (который после успешного перехода уменьшает значение переменной на единицу).
  Тогда новому условному оператору соответсвуют две дроби (по одной под каждый переход, в таком порядке): $\frac{13}{7\cdot2}$ и $\frac{17}{7}$.
- ```
  23. stop
  ```
  Кодируем как $\frac{1}{23}$. Это значит, что мы остановимся, так как для выполнения какой-то строчки мы должны поделить на простое, строго большее 5 (так как в знаменателе каждой дроби есть номер строки), а после обработки дроби $\frac{1}{23}$ в памяти останутся только переменные ($m = 2^a3^b5^c$).
  ```

Итого: мы построили по программе задачу FRACTRAN, т.е. написали программу на языке FRACTRAN. В том числе мы пострили m-сведение задачи останова к FRACTRAN: если бы решался FRACTRAN, то задача останова тоже бы решалась. ◀

1.7. Одноленточная машина Тьюринга

Имеется бесконечная лента, на которой изначально записан вход.

Σ -алфавит символов, которые используются на ленте. ▶ - символ начала ленты, после входа в ячейках записаны пробелы (бесконечное число).

Есть каретка, которая всегда указывает на какую-нибудь ячейку, изначально она указывает на первый символ после начала ленты.

Q — конечное множество состояний. q_0 — начальное состояние, q_f — конечное состояние.

Есть правила переходов:

$$\delta : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\leftarrow, \rightarrow, \cdot\}$$

По символу под картекой и текущему состоянию мы пишем какой-то новый символ, переходим в новое состояние и можем опционально сдвинуть каретку ($\cdot$ — стоять на месте, стрелочки — движение).

Результат работы — то, что после завершения программы написано на ленте до бесконечного числа пробелов.

Пример 1.7.1.

1. Хотим преобразовать `ВХОД` $\longrightarrow$ `УХОД`

   Правило перехода: $(q_0, \texttt{В}) \longrightarrow (q_f, \texttt{У}, \rightarrow)$.

2. Хотим к числу, которое нам дано в двоичной записи, прибавить 1: 11001 + 1 = 00101 (то есть записано оно на ленте справа налево).

   - $(q_0,\ 0) \longrightarrow (q_f,\ 1,\ \cdot)$
   - $(q_0,\ 1) \longrightarrow (q_0,\ 0,\ \rightarrow)$
   - $(q_0,\ \text{␣}) \longrightarrow (q_f,\ 1,\ \cdot)$

3. Теперь число двоичное, но записано слева направо. И тоже хотим прибавить 1: 10011 + 1 = 10100

   - $(q_0,\ 0/1) \longrightarrow (q_0,\ 0/1,\ \rightarrow)$
   - $(q_0,\ \text{␣}) \longrightarrow (q_1,\ \text{␣},\ \leftarrow)$
   - $(q_1,\ 1) \longrightarrow (q_1,\ 0,\ \rightarrow)$
   - $(q_1,\ 0) \longrightarrow (q_f,\ 1,\ \cdot)$
   - $(q_1,\ \blacktriangleright) \longrightarrow (q_2,\ \blacktriangleright,\ \rightarrow)$
   - $(q_2,\ 0) \longrightarrow (q_3,\ 1,\ \rightarrow)$
   - $(q_3,\ 0/1) \longrightarrow (q_3,\ 0/1,\ \rightarrow)$
   - $(q_3,\ \text{␣}) \longrightarrow (q_f,\ 0,\ \cdot)$

1.8. Тезис Чёрча

Алгоритм и машина Тьюринга — одно и то же.

Теорема 1.8.1. Если функция f вычисляется с помощью программы с конечным количеством переменных, то она вычисляется и с помощью машины Тьюринга.

▶ Доказательство относительно нестрого. Алфавитом будет $\{0, 1, \#, \sqcup\}$. На ленте мы будем записывать наши переменные в двоичном виде (от младших битов к старшим), разделяя их #: #a#b#c#…#

Состоянием машины Тьюринга будет текущая строка программы (коих конечное) и описание текущего действия, всего конечное число состояний.

1. `goto` и `stop` — это просто переход из одного состояния в другое
2. `if` — это надо съехать кареткой до начала ленты, потом «отсчитать» от начала ленты нужное (константное) число символов `#`, найдя таким образом нужную переменную. После этого остаётся лишь проверить, что она больше нуля (т.е. имеет хотя бы одну единицу в записи). Если дошли до следующей `#`, то делаем один `goto`, иначе — другой.
3. `a++` и `a--` выражаются похожим образом: сначала мы находим нужную переменную, потом прибавляем единицу. Проблема может возникнуть, если у нас увеличилась длина числа, а места не ленте нет. Тогда надо перезаписать ограничивающую нас `#` и потом «сдвинуть» остаток ленты (для этого нам хватит конечного числа состояний — машине надо помнить только текущий «сдвигаемый» символ).

◀

Теорема 1.8.2. Если функция вычисляется на машине Тюринга, то она вычисляется и с помощью программмы с конечным количеством переменных.

▶

Def 1.8.1. *Конфигурация* машины Тьюринга: (текущее состояние, лента левее головки, символ под головкой, лента правее головки).

Ленту левее и ленту правее можно реализовать через стек, а текущее состояние и символ под головкой — просто как две переменные.

Мы хотим представлять стек в виде одного числа. Если в алфавите k символов, то возьмём систему счисления по основанию k. Стек: $(a_1, a_2, \dots, a_l) \longrightarrow n = a_1 + ka_2 + \cdots + k^l a_l$. Верхний элемент: $n \bmod k$; удалить верхний элемент: $n \longrightarrow \lfloor \frac{n}{k} \rfloor$; добавить элемент: $x + nk$.

Таким образом получили 4 переменных, описывающих состояние МТ.

Проблемы (решаемые):

1. Если стек правее ленты пустой, добавить пробел
2. Преобразование входов и выходов.

◀

Def 1.8.2. *Универсальная машина Тьюринга*: $U(M, x)$, где M — описание некоторой машины Тьюринга, x — её вход. Тогда $U(M, x)$ запускает M на входе x.

Лента в процессе работы $U(M, x)$ выглядит примерно так: #*описание M*#*состояние M*#*лента M (вход)…*

Однако тут у нас скорость работы может замедлиться очень сильно (чем дальше каретка на ленте, тем медленнее переход). Чтобы этого не было (и было лишь константное замедление работы), можно таскать описание и состояние с собой по ленте. Например, при сдвиге каретки направо в самом начале работы из ленты

#*описание* M#*состояние* M#ВХОД

получится

B#*описание* M#*состояние* M#ХОД

Тогда чтобы посмотреть состояние и описание M нам потребуется сделать лишь константное число шагов по ленте.

1.9. Ассоциативное исчисление

Еще одна задача, которая окажется нерешаемой.

Def 1.9.1. *Одностороннее ассоциативное исчисление*: пусть Σ^* - множество слов алфавита Σ. Еще есть множество правил (односторонних) $x_1 \to y_1$, $x_2 \to y_2$, ..., $x_k \to y_k$, где $x_i, y_i \in \Sigma^*$.

Можно ли получить из строки a строку b? Разрешается брать подстроку a, равную некоторому x_i и заменять её на соответствующую y_i.

Def 1.9.2. Двустороннее ассоциативное исчисление: как одностороннее, только стрелочки в обе стороны. Частный случай одностороннего.

Пример 1.9.1. Правила:

$$\begin{cases} \texttt{AB} \longleftrightarrow \texttt{BA} \\ \texttt{BC} \longrightarrow \texttt{BB} \\ \texttt{C} \longrightarrow \texttt{CC} \end{cases}$$

Можно ли из ABAC получить ABBAC? Можно:

1. ABAC →AABC
2. AABC →AABCC
3. AABCC →AABBC
4. AABBC →ABABC
5. ABABC →ABBAC

Пример 1.9.2. Правила:

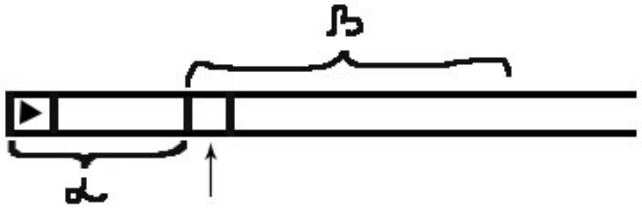
$$\begin{cases} \texttt{MAM} \longleftrightarrow \texttt{AA} \\ \texttt{AM} \longleftrightarrow \texttt{MAA} \\ \texttt{MM} \longleftrightarrow \texttt{MAM} \end{cases}$$

Можно ли из MAM получить AMA?

Оказывается, что нет алгоритма, решающего эту задачу (на вход даются правила и два слова).

Теорема 1.9.1. Выводимость в одностороннем ассоциативном исчислении алгоритмически невычислима.

▶ Сведём задачу останова к нашей задаче. Пусть есть машина Тьюринга M, хотим узнать, остановится ли она на входе x? Закодируем её строкой: $[\alpha\tilde{q}\beta]$, где $\tilde{q}$ — текущее состояние МТ, а α и β нарисованы ниже:



Алфавит положим равным объединению алфавитов МТ, множества её состояний, а также символов `[`, `]` и $*$. Теперь напишем правила, соответствующие переходам МТ. Идея такова: любой шаг машины Тьюринга изменяет её состояние только в районе $\tilde{q}$, поэтому можно написать правила замены подстрок, содержащих $\tilde{q}$. Мы напишем их таким образом, чтобы в каждый момент можно было применить ровно одно правило. Также мы добавим специальный символ $*$, который будет обозначать «мы достигли конечного состояния в машине», целью будет этот символ вывести.

- Правила на случай, если в β есть что-то, кроме пробелов:

Шаг машины Тюринга:	$(\tilde{q}, c) \longrightarrow (\widetilde{q'}, b, \rightarrow)$	$(\tilde{q}, c) \longrightarrow (\widetilde{q'}, b, \cdot)$
Строковая интерпретация:	$\tilde{q}c \longrightarrow b\widetilde{q'}$	$\tilde{q}c \longrightarrow \widetilde{q'}b$

  А также для всех x, кроме `[`:

Шаг машины Тюринга:	$(\tilde{q}, c) \longrightarrow (\widetilde{q'}, b, \leftarrow)$
Строковая интерпретация:	$x\tilde{q}c \longrightarrow \widetilde{q'}xb$

- Если в β только пробелы, то надо аккуратнее разбираться с правой границей:

Шаг машины Тюринга:	$(\tilde{q}, \sqcup) \longrightarrow (\widetilde{q'}, b, \rightarrow)$	$(\tilde{q}, \sqcup) \longrightarrow (\widetilde{q'}, b, \cdot)$
Строковая интерпретация:	$\tilde{q}\texttt{]} \longrightarrow b\widetilde{q'}\texttt{]}$	$\tilde{q}\texttt{]} \longrightarrow \widetilde{q'}b\texttt{]}$

  А также для всех x, кроме `[`:

Шаг машины Тюринга:	$(\tilde{q}, \sqcup) \longrightarrow (\widetilde{q'}, b, \leftarrow)$
Строковая интерпретация:	$x\tilde{q}\texttt{]} \longrightarrow \widetilde{q'}xb\texttt{]}$

Таким образом на каждый переход в машине Тьюринга мы добавляем два правила (на случай, если β пусто, и на случай, если непусто).

Тогда M остановится на $x \iff$ из $[q_0x] \to$ можно вывести $*$, где для $*$ есть такие правила вывода ($*$ может появиться только из конечного состояния и, если она появилась, то мы можем «съесть» все остальные символы):

- $q_f \to *$
- $*a \to *$ для всех символов a
- $b* \to *$ для всех символов b

Таким образом, мы построили сведение. А задача останова неразрешима. Если бы умели решать ассоциативное исчисление, то задачу останова тоже. ◀

Теорема 1.9.2. Двустороннее ассоциативное исчисление также неразрешима.

▶ В правила из предыдущего доказательства к каждой стрелке добавим обратную. Покажем, что если существует путь в двухстороннем исчислении, то в одностороннем тоже (для наших конкретных правил). В самом деле, пусть имеется какой-то путь, заканчивающийся в $*$. В $*$ по обратной стрелке не прийти, значит, последняя стрелка — «прямая».

Теперь рассмотрим *кратчайший* путь в $*$. Пусть в нём есть обратные стрелки ($\leftarrow$). Мы точно знаем, что последняя стрелка прямая ($\rightarrow$) Тогда где-то встречается сначала обратная стрелка ($B \leftarrow A$), а потом — прямая ($B \to C$). Заметим, что $A = C$, так как из каждой B у нас направо может идти не более одной стрелки (мы так строили правила). Значит, эти две стрелки можно сократить и получить путь меньшей длины, что противоречит предположению о его минимальной длине. ◀

Интерпретация в теории групп: есть полугруппа, заданная соотношениями $x_1 = y_1$, $x_2 = y_2$, ..., $x_k = y_k$ (где $x_i y_i$ — какие-то последовательности элементов в произведении). Тогда сравнение элементов a и b алгоритмически неразрешимо.

1.10. Теорема Успенского-Райса

Пусть A, B — алгоритмы.

Def 1.10.1. A ~ B (*эквивалентны*), если для всех x:

$$\begin{cases} A(x) \text{ остановится } \iff B(x) \text{ остановится} \\ A(x) \text{ остановится } \Rightarrow A(x) = B(x) \end{cases}$$

То есть мы не можем отличить A и B, смотря только на их поведение на разных входах.

Аналогичное обозначение для номеров: $n \equiv m \iff \langle n \rangle \sim \langle m \rangle$

Def 1.10.2. Пусть $S \subseteq \mathcal{N}$, тогда S называется *инвариантным*, если:

$$\forall a \in S, \forall b \in \mathcal{N} \setminus S \colon a \not\equiv b$$

То есть если a и b эквивалентны, то они либо оба лежат в S, либо оба там не лежат.

Замечание 1.10.1. Например, в будущем мы будем рассматривать множества алгоритмом, обладающих каким-то свойством. Если это свойство зависит только от поведения алгоритма на входах, то такие множества будут инвариантны.

Замечание 1.10.2. S инвариантно $\iff$ $\mathcal{N} \setminus S$ инвариантно.

Теорема 1.10.1 (Успенский-Райс)**.** Если S инвариантно, $S \neq \emptyset$, $S \neq \mathcal{N}$, то S неразрешимо.

▶ От противного. Пусть S разрешимо и нетривиально. Пусть Λ - алгоритм, который не останавливается ни на каком входе. Не умаляя общности можно считать, что $\#\Lambda \in S$. Возьмём какой-нибудь номер алгоритма $a \in \bar{S}$.

Возьмём какое-нибудь перечислимое и неразрешимое множество W (например, область определения диагональной функции из доказательства теоремы 1.3.1). Построим алгоритм $V(n, x)$, он запускает полуразрешающий алгоритм для W на входе n и, если тот остановится, запускает $\langle a \rangle(x)$ и выдаёт результат. На самом деле, у нас в каком-то смысле есть много алгоритмов (если зафиксировать первый аргумент): $V_1(x) = V(1, x), V_2(x) = V(2, x), \ldots$.

Теперь посмотрим на номера этих алгоритмов, чему они эквивалентны и где лежат:

$$\begin{cases} n \in W \Rightarrow \#V_n \equiv a \overset{a \in \bar{S}}{\Longrightarrow} \#V_n \in \bar{S} \\ n \notin W \Rightarrow \#V_n \equiv \#\Lambda \overset{\#\Lambda \in S}{\Longrightarrow} \#V_n \in S \end{cases}$$

Теперь построим m-сведение множества $\bar{S}$ к множеству W ($W \leqslant_m \bar{S}$): $f(n) = \#V_n$. В самом деле:

$$n \in W \iff \#V_n \in \bar{S}$$

Так как W неразрешимо, то $\bar{S}$ тоже неразрешимо $\Rightarrow$ S неразрешимо. ◀

1.11. Теорема о неподвижной точке

Теорема 1.11.1 (о неподвижной точке, теорема Клини о рекурсии)**.** Для любой всюду определённой вычислимой функции h существует $m \in \mathcal{N}$ такая, что $h(m) \equiv m$.

▶ Будет позже, после следствий. ◀

Следствие 1.11.1.1. Теорема Успенского-Райса: пусть S инвариантно, $S \neq \emptyset$, $S \neq \mathcal{N}$, тогда S неразрешимо.

▶ Пусть S разрешимо. Возьмём произвольный $a \in S$ и $b \notin S$. По инвариантности S знаем, что $a \not\equiv b$. Построим всюду определённую функцию $h(x)$ (это можно сделать, так как мы предположили разрешимость S):

$$h(x) = \begin{cases} a, & x \notin S \\ b, & x \in S \end{cases}$$

По теореме о неподвижной точке существует такое m, что $h(m) \equiv m$. Два случая:

- $m \in S \Rightarrow h(m) = b$, но $b \notin S \Rightarrow h(m) \not\equiv m$
- $m \notin S \Rightarrow h(m) = a$, но $a \in S \Rightarrow h(m) \not\equiv m$

Противоречие. ◀

Следствие 1.11.1.2. Существует алгоритм, печатающий в точности свой номер.

▶ Положим $h(x) = \#(\texttt{print x})$. По теореме о неподвижной точке найдём такое m, что $h(m) \equiv m$. Тогда $\langle m \rangle \equiv \texttt{print m}$, то есть эквивалентен какому-то, печатающему m, то есть $\langle m \rangle$ печатает m. ◀

Замечание 1.11.1. Как можно описать программы, печатающие свой текст («квайны», quine): пусть есть команда `print2 x`, которая эквивалентна `print x, ' ''', x, '''` (то есть печатает x, потом пробел и кавычку, потом опять x, потом еще раз кавычку). Тогда можно написать квайн `print2 ''print2''`

Def 1.11.1. Вычислимая функция f *≡-продолжает* вычислимую функцию g до всюду определённой вычислимой, если:

1. f всюду определена
2. если $g(x)$ определена, то $g(x) \equiv f(x)$

Для доказательства теоремы 1.11.1 (о неподвижной точке) нам потребуется еще одна, вспомогательная (которая, разумеется, теоремой о неподвижной точке не пользуется):

Теорема 1.11.2. Любую вычислимую функцию g можно ≡-продолжить до всюду определённой вычислимой.

Замечание 1.11.2. У нас была похожая теорема 1.3.1 про диагональную функцию, и там мы показывали, что её нельзя доопределить. Отличие в том, что в этой теореме мы не требуем точного совпадения значений функций, а лишь эквивалентности этих значений.

▶ Пусть алгоритм B вычисляет g (и зацикливается на не-области определения). Тогда рассмотрим алгоритм $A(n, x) = \langle B(n) \rangle (x)$. Разумеется, при n не из области определения g алгоритм A_n всегда зацикливается, потому что не может вычислить $B(n)$.

Теперь рассмотрим функцию $f(n) = \#A_n$. Эта функция всюду определена. Более того, пусть $g(n)$ определена, тогда $f(n) \equiv \#A_n \equiv \#\langle g(n) \rangle \equiv g(n)$. Таким образом, f является искомой функцией. ◀

Теперь доказательство теоремы о неподвижной точке:

▶ Вспомним диагональную функцию $u(n) = \langle n \rangle(n)$. По теореме 1.11.2 найдём $\equiv$-продолжение u — функцию $g(n)$. У нас в условии фигурировала функция h (у которой мы ищем неподвижную точку), давайте рассмотрим функцию $t(n) = h(g(n)) = (h \circ g)(n)$. Это всюду определённая функция (так как и h, и g всюду определены). Теперь заметим, что:

$$u(\#t) = \langle \#t \rangle(\#t) = t(\#t) = h(g(\#t))$$

С другой стороны, так как g есть $\equiv$-продолжение u и $u(\#t)$ определена:

$$u(\#t) \equiv g(\#t)$$

Таким образом имеем $h(g(\#t)) \equiv g(\#t)$. Положив $m = g(\#t)$ получим неподвижную точку m для функции h, что и требовалось доказать. ◀

Другое доказательство, менее формальное:

▶ Возьмём какой-нибудь язык программирования и расширим его функциями:

- `getProgramText()` — возвращает текст текущей программы (как квайн)
- `execute(s)` — выполняет программу, текст которой передан, как параметр
- `computeH(s)` — вычисляет h в точке s (например, если туда передан текст программы, то в точке $\#s$). Это можно реализовать, так как h всюду вычислима.

Теперь пишем программу m:

```
s = getProgramText();
s = computeH(s);
execute(s);
```

То есть эта программа при запуске $m()$ применяёт h к самой себе и выполняет результатирующий алгоритм. С другой стороны, если запустить $h(m)()$, то результат будет тот же: h вычислит значение в m, а результат потом запустим. Таким образом, $h(m) \equiv m$, что и требовалось. ◀

1.12. Нумерации

Def 1.12.1. Вычислимая функция $U(n, x)$ называется *универсальной вычислимой функцией* или *нумерацией*, если для любой вычислимой f существует такое m, что $\forall x\colon f(x) = U(m, x)$.

Def 1.12.2. Универсальная вычислимая функция $U(n, x)$ называется *главной*, если для любой вычислимой функции $V(n, x)$ есть всюду определённая вычислимая S такая, что:

$$V(n, x) = U(S(n), x)$$

То есть существует способ «переформулировки программ» с любого языка V на язык U.

Пример 1.12.1. $U(n, x) = \langle n \rangle(x)$ (универсальный алгоритм) является универсальной вычислимой функцией.

▶ Возьмём какую-нибудь вычислимую функцию $V(n, x)$. Переобозначим $V(n, x) = V_n(x)$, V_n — тоже какие-то вычислимые функции. Положим $S(n) = \#V_n$. Тогда:

$$U(S(n), x) = U(\#V_n, x) = V_n(x) = V(n, x)$$

◀

Теорема 1.12.1 (о неподвижной точке в главной нумерации $W(n,x)$)**.** Пусть $h(x)$ — всюду определённая вычислимая функция. Тогда существует такое m, что $W(m,x) = W(h(m),x)$ для любого x.

Замечание 1.12.1. Для стандартной нумерации $U(n,x)$ это уже доказано в теореме 1.11.1. В самом деле: $a \equiv b \iff \forall x\colon U(a,x) = U(b,x)$

Тогда для доказательства теоремы достаточно доказать следующее утверждение:

Утверждение 1.12.1. Если теорема о неподвижной точке верна в какой-то главной нумерации U, то она верна и в другой главной нумерации W.

▶ Так как $U(n,x)$ главная, то есть всюду определённая g такая, что:

$$W(n,x) = U(g(n),x)$$

Аналогично, так как $W(n,x)$ главная, то есть всюду определённая s такая, что:

$$U(n,x) = W(s(n),x)$$

Теперь возьмём $h(x)$ из условия теоремы. Рассмотрим $g \circ h \circ s$ — это всюду определённая функция. Так как теорема о неподвижной точке верна для главной нумерации U, найдём m такое, что для любого x:

$$\begin{aligned} U(m,x) &= U((g\circ h\circ s)(m),x) = U(g(h(s(m))),x) = W(h(s(m)),x)\\ U(m,x) &= W(s(m),x)\\ W(h(s(m)),x) &= W(s(m),x) \end{aligned}$$

Таким образом, $s(m)$ — неподвижная точка для h в нумерации главной нумерации W. ◀

1.13. Арифметичность предикатов

1.13.1. Напоминание

Сначала определим предикатные формулы (как в прошлом году).

У нас есть множества предикатных символов ($\mathcal{P}$) и функциональных символов ($\mathcal{F}$) с фиксированными арностями (у разных символов арности могут отличаться). Например, в арифметике было $\mathcal{P} = \{=\}$, $\mathcal{F} = \{*, +\}$.

𝔇𝔢𝔣 1.13.1. Терм — либо переменная (x, y, a_3), либо $f(t_1, t_2, \dots, t_k)$, где $f \in \mathcal{F}$, а t_i — термы.

То есть терм — это формула из переменных и функциональных символов, например: $(x+y)\cdot z+x$.

𝔇𝔢𝔣 1.13.2. Атомарная формула — это $p(t_1, \dots, t_k)$, где $p \in \mathcal{P}$, а t_i — термы.

То есть это условие на термы, например: $x + y = z \cdot \alpha$.

𝔇𝔢𝔣 1.13.3. Предикатная формула — это одно из нескольких:

- Атомарная формула
- Комбинация предикатных формул A и B: $A \vee B$, $A \wedge B$, $\neg A$, (A), $A \Rightarrow B$
- Предикатная формула с добавленным квантора по переменной x: $\forall x\colon A$ или $\exists x\colon A$

Теперь придаём смысл формулам.

Def 1.13.4. Пусть есть *носитель интерпретации* — множество M, для каждого предикатного символа арности k есть функция $M^k \to \{0,1\}$, а для каждого функционального символа арности l есть функция $M^k \to M$.

Тогда мы можем естественным образом «проверить» истинность предикатной формулы — это и будет *интерпретацией* формулы.

Def 1.13.5. Предикат $p\colon M^k \to \{0,1\}$ является *выразимым*, если есть некоторая предикатная формула P с k свободными переменными (то есть не участвующими в кванторах), которая задаёт в точности этот предикат (то есть значения предиката и формулы совпадают на всех значениях переменных).

Def 1.13.6. Арифметический предикат — это предикат, выразимый в арифметике. Носителем в арифметике является $\mathcal{N} = \mathbb{N} \cup \{0\}$, а символами $\mathcal{P} = \{=\}$, $\mathcal{F} = \{*, +\}$ с естественными смыслом в неотрицательных целых числах.

Примеры выразимых предикатов:

$x = 0$	$\forall y\colon y + x = y$
$x = 1$	$\forall y\colon y \cdot x = y$
$x = 238$	$\forall y\colon (y = 1) \Rightarrow x = \underbrace{y + y + \cdots + y}_{238 \text{ слагаемых}}$
$x \geqslant y$	$\exists z\colon x = y + z$
$x > y$	$\exists z\colon \neg(z = 0) \wedge x = y + z$
$x \bmod y = z$	$\exists k\colon (x = y \cdot k + z) \wedge (z < y)$
x— простое	$(\forall z\colon (x \vdots z) \Rightarrow (z = x) \vee (z = 1)) \wedge \neg(x = 1)$

1.13.2. Бета-функция Гёделя

Давайте посмотрим на предикат $x = y^z$. Наивное выражение этого предиката может выглядеть вот так:

$$\exists t_1, t_2, \ldots t_z (t_1 = y \wedge t_z = x \wedge \forall i\colon ((1 \leqslant i < z) \Rightarrow t_{i+1} = t_i y)$$

Это мы просто задали последовательность t_i, начинающуюся с y, заканчивающуюся на x и с фиксированным отношением между соседними членами. Но, вообще говоря, так писать нельзя — это выражение не является строкой, так как его длина неопределена и зависит от z.

Год назад мы боролись с этой проблемой, кодируя множества натуральными числами и вводя предикат «число принаджлежит множеству». Сейчас рассмотрим другой подход, подход Гёделя. Этот подход и называется β-функцией Гёделя. Сначала нужно несколько лемм:

Лемма 1.13.1. $\forall k \in \mathbb{N}\colon \exists$ сколь угодно большое натуральное b такое, что $b + 1$, $2b + 1$, ..., $kb + 1$ являются попарно взаимно простыми.

▶ Возьмём любое $b \vdots k!$ и покажем, что оно подходит.

В самом деле, если это не так, то есть некоторые $i, j \in [1, k]$ такие, что $(ib + 1, jb + 1) \neq 1$. Тогда пусть p — такое простое, что $ib + 1, jb + 1 \vdots p$, откуда $|i - j| b \vdots p$. Заметим, что b не может делиться на p (иначе бы $ib + 1$ не делилось на p).

Таким образом имеем $|i - j| \vdots p$. Так как $|i - j| \leqslant k$, то $k! \vdots |i - j| \Rightarrow k! \vdots p \Rightarrow b \vdots p$. Противоречие. ◀

Лемма 1.13.2. Для любой последовательности x_0, x_1, ..., $x_n \in \mathbb{N}$ найдутся такие $a, b \in \mathbb{N}$, что:

$$\begin{aligned} x_0 &\equiv a \mod (b + 1) \\ x_1 &\equiv a \mod (2b + 1) \\ &\vdots\vdots \\ x_n &\equiv a \mod ((n + 1)b + 1) \end{aligned}$$

▶ По лемме 1.13.1 выберем такое $b > \max x_i$, что все числа $(b+1)$, $...((n+1)b+1)$ попарно взаимно просты. Тогда по китайской теореме об остатках существует решение a для системы:

$$\begin{aligned} a &\equiv x_0 \mod (b+1) \\ a &\equiv x_1 \mod (2b+1) \\ &\vdots \\ a &\equiv x_n \mod ((n+1)b+1) \end{aligned}$$

Оно и будет вторым требуемым числом. ◀

Возвращаясь к примеру $x = y^z$: теперь всего двумя числами можно закодировать последовательность $t_1, ..., t_z$. Вообще говоря, оператора mod у нас в строках-предикатах нет, но мы его выводили, можем подставить. Тогда можем выразить наш предикат таким образом: существуют a и b такие, что любые два соседних числа из соответствующей последовательности $t_1, ..., t_z$ ($t_k = a \bmod (kb+1)$) связаны выражением $t_{k+1} = t_k \cdot y$, причём $t_1 = y$, а $t_z = x$. При этом нам неважно, могли ли вообще a и b получиться в результате кодирования последовательности длины z — любая пара кодирует какую-то последовательность, но любая последовательность кодируется некоторой парой, биекция не требуется. На самом деле, даже неважно, какая именно длина у последовательности, задаваемой парой (a, b) — каждая пара задаёт некоторую бесконечную последовательность, а нас интересует только префикс фиксированной длины z.

Итого, записываем:

$$\begin{aligned} &\beta(a,b,k) = a \bmod (k \cdot b + 1) \\ &(x = y^z) \iff \exists a, b\colon (\beta(a,b,1) = y) \wedge (\beta(a,b,z) = x) \wedge \\ &\qquad (\forall k\colon ((1 \leqslant k) \wedge (k < z)) \Rightarrow (\beta(a,b,k) \cdot y = \beta(a,b,k+1))) \end{aligned}$$

1.13.3. Перечислимость и выразимость в арифметике

𝔇𝔢𝔣 1.13.7. Предикат $P\colon \mathbb{N} \to \{0,1\}$ является перечислимым, если $\{x \mid P(x) = 1\}$ является перечислимым.

𝔇𝔢𝔣 1.13.8. Предикат $P\colon \mathbb{N} \to \{0,1\}$ является разрешимым, если $\{x \mid P(x) = 1\}$ является разрешимым.

Теорема 1.13.1. Любой перечислимый предикат является арифметическим (выразим в арифметике).

▶ Сначала выберем удобную модель. Для машин Тьюринга можно, но гораздо удобнее будет оперировать программами с конечным числом переменных (они ведь уже оперируют с натуральными числами). Мы знаем, что так как P перечислим, то по одному из определний перечислимости существует полуразрешающий алгоритм, то есть программа с конечным числом переменных S такая, что $P(x) = 1 \iff S(x)$ останавливается.

Пусть имеется какая-то программа. Можно считать, что в ней ровно одна команда остановки программы (`stop`), потому что остальные можно заменить на `goto` к какому-нибудь `stop`. С концом программы тоже никаких проблем: в зависимости от того, что, как мы считаем, происходит в конце программы, можно делать разные вещи (бесконечный цикл или `goto` на `stop`).

Теперь давайте запишем следующее утверждение: существует некоторая последовательность состояний программы длины T, начинающаяся в изначальном состоянии и заканчивающаяся

в единственном `stop`, причём переходы между соседними состояниями корректны. Последовательность состояний — это просто несколько последовательностей, соответствующих переменным и номеру текущей команды. Запишем в «некорректной» форме (с неопределённой длиной строки), при помощи β-функции Гёделя от этого легко избавиться:

1. Существует последовательность состояний:
$$\exists T; \exists a_1^{(1)}, \dots, a_1^{(T)}; \exists a_2^{(1)}, \dots, a_2^{(T)}; \dots; \exists m^{(1)}, \dots, m^{(T)}:$$
2. Начинаем где надо (S_1 — номер первой строчки):
$$(a_1^{(1)} = x) \wedge (a_2^{(1)} = 0) \wedge \dots \wedge (a_m^{(1)=0}) \wedge (m^{(1)} = S_1)$$
3. И заканчиваем где надо:
$$m^{(T)} = S_{stop}$$
4. А также все шаги корректны:
$$\forall i: (i < T) \wedge (i \geqslant 1) \Rightarrow Step(a_1^{(i)}, a_2^{(i)}, \dots, a_k^{(i)}, m^{(i)}, a_1^{(i+1)}, a_2^{(i+1)}, \dots, a_k^{(i+1)}, m^{(i+1)},)$$

Осталось только написать предикат *Step*. Он является просто разбором случаев: берём логическое «и» условий для всех строк, для строки S_x условие выглядит как $(m^{(i)} = S_x) \Rightarrow (\dots)$, внутри скобок пишем условие на корректность перехода.

1. Если идёт изменение переменной, то оно выразимо в арифметике, остальные поменяться не должны, номер следующей строки тоже выразим.
2. Если идёт переход, то переменные не поменялись, номер следующей строки тоже выразим.
3. В случае с условным оператором вида `if (a > 0) then goto x else goto y` можно записать условие на постоянность значений всех переменных, а также:
$$((a^{(i)} > 0) \Rightarrow (m^{(i+1)} = x)) \wedge ((a^{(i)} = 0) \Rightarrow (m^{(i+1)} = y))$$

◀

Следствие 1.13.1.1. График любой вычислимой функции $f: A \to \mathbb{N}$ выразим в арифметике. Напоминание: $\Gamma_f = \{(x, f(x)) \mid x \in A\}$.

Следствие 1.13.1.2. Любой разрешимый предикат выразим в арифметике (просто потому что он перечислим).

1.14. Арифметическая иерархия

1.14.1. О предварённой форме

Def 1.14.1. Формулы φ_1 и φ_2 эквивалентны, если у них одинаковое количество свободных переменных, одинаково именованные свободные переменные и во всех интерпретациях и во всех оценках свободных переменных их значения совпадают.

Напоминание: оценка свободных переменных — это подстановка значений, а интерпретация — это когда мы задаём, какие у нас символы какие предикаты обозначают (например, что обозначает < и >).

Лемма 1.14.1 (о предварённой форме)**.** Для любой формулы можно построить эквивалентную ей, у которой все кванторы (Q_i) стоят в начале:
$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n: \varphi(x_1, \dots, x_n, x_{n+1}, \dots x_m)$$
(при этом в $\varphi(\dots)$ кванторов нет).

▶ Давайте научимся выносить очередной квантор в начало формулы.

- $\neg(\forall x\varphi)$ эквивалентно $\exists x(\neg\varphi)$.
- $\neg(\exists x\varphi)$ эквивалентно $\forall x(\neg\varphi)$.

А вот дальше сложнее, потому что у нас некоторые связные переменные могут совпадать по именам с другими связными или свободными, заменять $(\exists x\colon \varphi) \vee \psi$ на $\exists x\colon (\varphi \vee \psi)$ нельзя. Поэтому сначала надо переименовать все связные переменные так, чтобы имена вообще всех встречающихся переменных были попарно различны. А потом можно применять правила (первые два для отрицания уже были), вот еще одно:

$$(\exists x\varphi) \vee \psi \sim \exists x(\varphi \vee \psi)$$

Для других операций и кванторов аналогично. ◀

1.14.2. Иерархия Клини

Еще называется арифметической иерархией.

Мы классифицируем предикаты по разным классам, а точнее — в классы Σ_i и Π_i. Начинаем с $\Sigma_0 = \Pi_0$ — множество разрешимых предикатов.

Def 1.14.2. $P(x_1, \dots, x_m) \in \Sigma_1$, если существует разрешимый предикат $Q(y, x_1, \dots x_m)$ (то есть предикат из $\Sigma_0 = \Pi_0$) такой, что

$$\forall x_1, x_2, \dots x_m\colon P(x_1, \dots, x_m) = \exists y Q(y, x_1, \dots, x_m)$$

Замечание 1.14.1. Σ_1 есть множество перечислимых предикатов.

▶ В самом деле: для любого перечислимого есть алгоритм, который на шаге y выдаёт какой-то вход, на котором предикат верен, тогда этот алгоритм можно «обозначить» за предикат Q, который уже разрешим (ведь алгоритм выполняет не более y шагов). И в обратную сторону: если есть Q, то просто проверяем по очереди все y. ◀

Def 1.14.3. $P(x_1, \dots, x_m) \in \Pi_1$, если существует разрешимый предикат $Q(y, x_1, \dots x_m)$ (то есть предикат из $\Sigma_0 = \Pi_0$) такой, что

$$\forall x_1, x_2, \dots x_m\colon P(x_1, \dots, x_m) = \forall y Q(y, x_1, \dots, x_m)$$

Π_1 есть множество коперечислимых предикатов (у которых дополнение перечислимо).

Def 1.14.4. $P(x_1, \dots, x_m) \in \Sigma_{k+1}$ если $\exists Q(y, x_1, \dots, x_m) \in \Pi_k$ такой, что

$$\forall x_1, x_2, \dots x_m\colon P(x_1, \dots, x_m) = \exists y Q(y, x_1, \dots, x_m)$$

Аналогичное определение для Π_{k+1}:

Def 1.14.5. $P(x_1, \dots, x_m) \in \Pi_{k+1}$ если $\exists Q(y, x_1, \dots, x_m) \in \Sigma_k$ такой, что

$$\forall x_1, x_2, \dots x_m\colon P(x_1, \dots, x_m) = \forall y Q(y, x_1, \dots, x_m)$$

Можно определить нерекурсивно, там получится чередование кванторов:

- $P(x_1, \dots, x_m) \in \Sigma_k \iff$ есть разрешимый $Q(y_1, y_2, \dots, y_k, x_1, \dots, x_m)$ такой, что:

$$P(x_1, \dots, x_m) = \exists y_1 \forall y_2 \exists y_3 \dots Q_k y_k\colon Q(y_1, y_2, \dots, y_k, x_1, \dots, x_m)$$

- $P(x_1, \dots, x_m) \in \Pi_k \iff$ есть разрешимый $Q(y_1, y_2, \dots, y_k, x_1, \dots, x_m)$ такой, что:

$$P(x_1, \dots, x_m) = \forall y_1 \exists y_2 \forall y_3 \dots Q_k y_k \colon Q(y_1, y_2, \dots, y_k, x_1, \dots, x_m)$$

Пока что это всё является непонятными формальными значками, это нормально. Скоро разберёмся, какой у всего этого смысл. Начнём с каких-нибудь свойств:

1.
$$\forall k \geqslant 0 \colon (\Sigma_k \cup \Pi_k) \subseteq \Sigma_{k+1} \cap \Pi_{k+1}$$

Надо показать, что каждое из множеств слева лежит в каждом из множеств справа.

▶
- $\Sigma_k \subseteq \Pi_{k+1}$ — добавляем фиктивную переменную в начало предиката: если $P(x) \in \Sigma_k$, то:
$$(\forall y \colon P(x)) \in \Pi_{k+1}$$
- $\Sigma_k \subseteq \Sigma_{k+1}$ — можно добавить фиктивный квантор в конец:

$$\begin{aligned} &P(x_1, x_2, \dots, x_n) \in \Sigma_k \\ &P(x_1, x_2, \dots, x_n) = \exists y_1 \forall y_2 \dots Q_k y_k \colon Q(y_1, \dots, y_k, x_1, \dots, x_n) = \\ &\qquad = \exists y_1 \forall y_2 \dots Q_k y_k Q_{k+1} y_{k+1} \colon Q(y_1, \dots, y_k, x_1, \dots, x_n) \in \Sigma_{k+1} \end{aligned}$$

Для вложений Π_k всё симметрично. ◀

2. Π_k — это множество дополнений предикатов из Σ_k.

▶ Можно по индукции и k, можно просто взять отрицание предиката, кванторы поменяются на противоположные. ◀

3. Любой арифметический предикат содержится в некотором уровне иерархии.

▶ Берём соответствующую ему арифметическую формулу, приводим в предварённую форму (выносим все кванторы вперёд). Формула, конечно, поменяется, но ничего страшного: у нас каждый предикат может иметь несколько форм записи, это нормально.
Проверку истинности безкванторного остатка можно сделать алгоритмом (так как там только арифметические и логические операции). Получили почти определение класса, только без чередования кванторов. Можно либо добавить фиктивных кванторов над фиктивными неиспользуемыми переменными, либо «запихнуть» в одну переменную несколько. ◀

4. Любой предикат из Σ_k арифметический.

▶ У нас есть какая-то формула вида

$$\exists y_1 \forall y_2 \dots y_i \colon Q(y_1, \dots, y_i, x_1, \dots, x_n)$$

Формула Q разрешима и, соответственно, арифметическая. А в начале у нас просто идёт куча кванторов, которые мы можем дописывать в арифметическое выражение. ◀

5. Σ_i и Π_i для $i \geqslant 0$ «замкнуты» (в одну сторону) относительно m-сведения. Напоминание: $A \leqslant_m B$ (B m-сводится к A) $\iff \exists f$ (всюду определённая вычислимая функция) такая, что $x \in A \iff f(x) \in B$.

Утверждение 1.14.1. Пусть $P_1(x_1, \dots, x_n) \in \Sigma_i$ и $P_2 \leqslant_m P_1$. Тогда $P_2 \in \Sigma_i$.

▶ Так как $P_1 \in \Sigma_i$, то существует разрешимый предикат Q и для всех $x_1, \dots, x_n$ верно:

$$P_1(x_1, \dots, x_n) = 1 \iff \exists y_1, \forall y_2, \dots, y_i \colon Q(y_1, y_2, \dots, y_i, x_1, \dots x_n)$$

Так как $P_2 \leqslant_m P_1$, то есть f из определения m-сводимости. Давайте представим P_2:

$$P_2(z_1, \dots, z_n) = P_1(f(z_1, \dots, z_n))$$
$$P_2(z_1, \dots, z_n) = \exists y_1, \forall y_2, \dots, y_i \colon Q(y_1, y_2, \dots, y_i, f(z_1, \dots, z_n))$$

Q был разрешим, мы кусок аргументов заменили на что-то, вычисляемое всегда останавливающимся алгоритмом. Значит, результатирующий предикат тоже разрешим. Таким образом, P_2 имеет ровно такой вид, какой надо для попадания в Σ_i. ◀

6.

Def 1.14.6. Пусть X — множество k-местных предикатов, $(k+1)$-местный предикат u называется универсальным для x, если:

$$\forall P \in X \colon \exists p \in \mathcal{N} \colon \forall x_1, \dots, x_n \in \mathcal{N} \colon P(x_1, \dots, x_n) = u(p, x_1, \dots, x_n)$$

То есть если любой предикат из X можно получить из u фиксацией первого аргумента.

Утверждение 1.14.2. Для любого $i \geqslant 1$ и $k \in \mathcal{N}$ существует универсальный предикат для k-местных предикатов из Σ_i, который сам лежит в Σ_i. Аналогично для Π_i.

▶ **База:** Для простоты считаем $k = 1$ (для двухместных и более предикатов всё точно так же). Найдём универсальный предикат для Σ_1, например:

$$U(n, x) = \begin{cases} 1, & \text{если } \langle n \rangle(x) \text{ останавливается} \\ 0, & \text{иначе} \end{cases}$$

В самом деле, $U(n, x) \in \Sigma_1$, так как он полуразрешим. Теперь покажем, что он универсален. Возьмём $P \in \Sigma_1$. Пусть полуразрешающий алгоритм для P имеет номер p. Тогда $P(x) = 1$ тогда и только тогда, когда $\langle p \rangle$ останавливается $\iff U(p, x) = 1$. Значит U универсален для Σ_1.
Универсальным для Π_1 будет отрицание этого предиката ($\neg U(p, x)$).

Переход $i \to i+1$: Рассмотрим какой-нибудь $P \in \Sigma_{i+1}$, по определению существует предикат $Q \in \Pi_i$ такой, что для всех x:

$$P(x) = 1 \iff \exists y \colon Q(y, x) = 1$$

Так как $Q \in \Pi_i$, то можем взять универсальный предикат U_{Π_i} для двухместных из Π_i (если P больше, чем одноместный, то тут возьмёся универсальный предикат для трёхместных и так далее). Значит, существует $q \in \mathcal{N}$ такое, что $Q(y, x) = U_{\Pi_i}(q, y, x)$. Тогда для всех x:

$$P(x) = 1 \iff \exists y \colon U_{\Pi_i}(q, y, x)$$

Теперь выпишем универсальный предикат для Σ_{i+1}:

$$U_{\Sigma_{i+1}}(p, x) = 1 \iff \exists y \colon U_{\Pi_i}(p, y, x)$$

Осталось проверить, что $U_{\Sigma_{i+1}} \in \Sigma_{i+1}$, что несложно: он имеет вид $\exists y \colon X$, где $X \in \Pi_i$, что и требуется.

◀

Упражнение 1.14.1. Для $i = 0$ свойство неверно.

Упражнение 1.14.2. Выписать универсальный предикат явно.

7. Для любого $i \geqslant 0$ универсальный предикат для одноместных из Σ_i не содержится в Π_i. Тут уже одноместность важна (для более высоких порядков вроде сложнее получается).

   ▶ Пусть U_{Σ_i} — универсальный предикат для Σ_i, лежащий в Π_i. Возьмём отрицание T этого предиката:
   $$T(n, x) = \neg U_{\Sigma_i}(n, x)$$
   (a) Так как $U_{\Sigma_i} \in \Pi_i$ (по предположению), то $T \in \Sigma_i$.
   (b) Предикат $t(x) = T(x, x)$ тоже лежит в Σ_i, так как мы просто выполнили подстановку.
   (c) Тогда давайте выразим t через наш универсальный предикат. Существует $k \in \mathcal{N}$ такое, что для всех x:
   $$t(x) = U_{\Sigma_i}(k, x)$$
   (d) С одной стороны, $t(k) = U_{\Sigma_i}(k, k)$. С другой стороны, это $\neg U_{\Sigma_i}(k, k)$. Противоречие. ◀

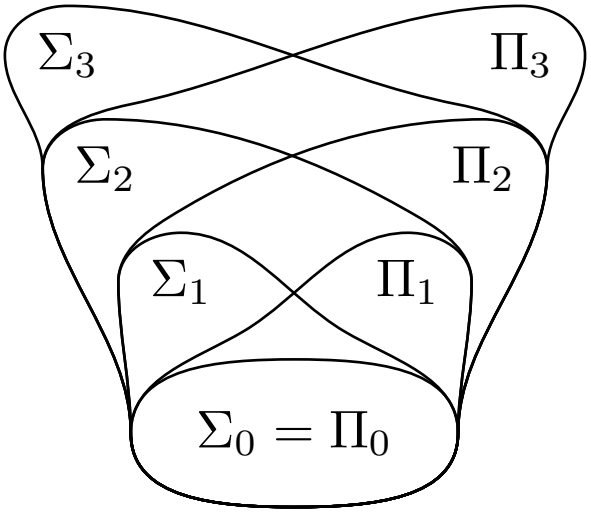
8. Для $i \geqslant 1$ имеем $\Sigma_i \neq \Pi_i$.

   ▶ Есть универсальный предикат U_{Σ_i}, лежащий в Σ_i, не лежащий в Π_i. ◀

9. $\Sigma_{i+1} \supsetneq \Sigma_i$.

   ▶ $\Sigma_{i+1} \supseteq \Pi_i \supsetneq \Pi_i \setminus \Sigma_i \neq \emptyset$. ◀

Картинка с иерархией



1.14.3. Теоремы Тарского и Гёделя

Обозначим за $\mathcal{T}$ (от слова «truth») множество номеров замкнутых формул в арифметике, которые истинны. Например, вот такая арифтметическая формула лежит в $\mathcal{T}$ (Великая теорема Ферма́):
$$\forall x, y, z, n\colon ((n \geqslant 3 \wedge x > 0 \wedge y > 0) \Rightarrow x^n + y^n \neq z^n)$$

Теорема 1.14.1 (Тарского)**.** «Истинность невыразима». Множество $\mathcal{T}$ невыразимо в арифметике.

▶ От противного. Пусть $\mathcal{T}$ выразим в арифметике предикатом T. Тогда $T \in \Sigma_k$ для некоторого k (если содержится в некотором Π_x, то содержится и в Σ_{x+1}). Докажем, что любой арифметический предикат P m-сводится к T ($P \leqslant_m T$).

Возьмём какой-нибудь предикат $P(x)$, который задаётся формулой $\varphi(x)$. Возьмём функцию $f(x) = \#\varphi(x)$, то есть номер формулы φ, в которой единственный аргумент заменили на константу x, получили формулу с нулём аргументов, взяли её номер.

Замечание 1.14.2. Заменять аргумент можно, например, конструкцией вида $\forall y\colon (y = x) \Rightarrow \varphi(y)$, если у нас в формулах нет понятия «константа в чистом виде».

Получили следующее:

$$P(x) = 1 \iff \varphi(x) = 1 \iff T(f(x)) = 1 \iff T(\#\varphi(x)) = 1$$

Что есть в точности определение m-сводимости P к T.

Теперь возьмём произвольный $P \in \Sigma_{k+1}$. Так как $P \leqslant_m T$, то $P \in \Sigma_k$, так как Σ_k замкнуто относительно m-сведения. То есть $\Sigma_k = \Sigma_{k+1}$, противоречие. ◀

Следствие 1.14.1.1 (Первая теорема Гёделя о неполноте). $\mathcal{T}$ не является перечислимым.

▶ Если бы $\mathcal{T}$ было перечислимым, то оно было бы и выразимым в арифметике, что неверно. ◀

Пояснение из Computer Science

Предыдущая формулировка несколько отличается от того, что обычно слышат перед курсом логики. Обычно слышат что-то вроде «нет системы доказательств, в которой можно доказать все утверждения про натуральные числа».

Def 1.14.7. Система доказательств для множества A (подмножество каких-нибудь строк, «утверждений») — это всюду определённый алгоритм V (от слова «verify»). Параметра два: x (утверждение) и w («witness», собственно, доказательство). Свойства:

Корректность: $V(x, w) = 1 \Rightarrow x \in A$

Полнота $\forall x \in A\colon \exists w\colon V(x, w) = 1$

Замечание 1.14.3. Это определение из Computer Science, обобщающее похожие утверждения из логики.

Замечание 1.14.4. Например, в матанализе мы обычно доказываем, что какое-то утверждение («теорема») лежит в множестве верных утверждений.

Теорема 1.14.2. Множество A имеет систему доказательств $\iff$ A перечислимо.

▶ **$\Rightarrow$:** Предъявим полуразрешающий алгоритм для A (тогда A перечислимо). Называется «алгоритм Британского музея» (потому что в этом музее есть всё, в том числе доказательства любой теоремы). Просто перебираем все w, для каждого запускаем V (который всегда завершается).

$\Leftarrow$: Пусть A перечислимо, B — полуразрешимый алгоритм для A. Тогда доказательством для x будет являться число w — сколько шагов алгоритма B надо выполнить, чтобы B завершился и сказал «$x \in A$»:

$$V(x, w) = \begin{cases} 1, & \text{если } B(x) \text{ останавливется после } w \text{ шагов} \\ 0, & \text{иначе} \end{cases}$$

◀

Следствие 1.14.2.1. Так как $\mathcal{T}$ неперечислимо, то для него нет системы доказательств.

Альтернативное доказательство теоремы Гёделя

▶ Пусть $\overline{W} = \{n \mid \langle n \rangle(n)$ не останавливается$\}$ (а $W = \overline{\overline{W}}$). Мы знаем, что W неразрешимо, но перечислимо, значит, $\overline{W}$ неперечислимо (иначе бы W было разрешимым по теореме Поста).

Хотим доказать, что $\overline{W}$ m-сводится к $\mathcal{T}$ ($\overline{W} \leqslant_m \mathcal{T}$). тогда будет следовать, что $\mathcal{T}$ неперечислимо, так как $\overline{W}$ неперечислимо.

Возьмём функцию:

$$f(n) = \#(\langle n \rangle(n) \text{ не останавливается})$$

Покажем, что справа стоит какая-то формула. В самом деле, «$\langle n \rangle(n)$ не останавливается» есть не что иное, как предикат «для любого t алгоритм $\langle n \rangle(n)$ не останавливается за t шагов», вторая часть выражается в арифметике (так как является разрешимым предикатом), квантор приписать тоже можем. ◀

1.15. Рекурсивные функции

Это еще один способ определить вычислимые функции, похожий на функциональные языки программирования. По сути — еще одна формальная модель вычислений.

Def 1.15.1. Операция подстановки: пусть $h(x_1, \dots, x_k)$ — функция, и $g_1, g_2, \dots, g_k \colon \mathcal{N}^n \to \mathcal{N} \cup \{\perp\}$, то можно составить функцию

$$h(g_1(x_1, \dots, x_n), g_2(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n))$$

Тут $\perp$ обозначает, что функция не определена.

Def 1.15.2. Операция примитивной рекурсии: пусть есть функции f, g, тогда можно определить h;

$$h(x_1, \dots, x_k, 0) = f(x_1, \dots, x_k)$$
$$h(x_1, \dots, x_k, y+1) = g(x_1, x_2, \dots, x_k, y, h(x_1, \dots, x_n, y))$$

Def 1.15.3. Примитивно рекурсивная функция — это функция, которая может быть получена при помощи правил подстановки и примитивной рекурсии из следующих базисных:

1. 0-константа
2. $S(x) = x + 1$
3. $\pi^i_k(x_1, \dots, x_k) = x_i$

Пример 1.15.1. $x + y$ можно определить примитивно рекурсивно:

$$x + 0 = x = \pi^1_1(x)$$
$$x + (y+1) = S(x+y)$$

Пример 1.15.2. $x \cdot y$ можно определить примитивно рекурсивно:

$$x \cdot 0 = 0$$
$$x \cdot (y+1) = (x \cdot y) + \pi^1_2(x, y) = (x \cdot y) + x$$

Формально проекция $\pi^1_2(x, y)$ нужна, но мы её дальше будем опускать.

Замечание 1.15.1. У нас, вообще говоря, функции могли выдавать значение $\perp$ («штопор»), а принимать не вход — не могли. Писать композицию в таком случае неаккуратно. Поэтому мы считаем, что $\perp$ у нас на случай «значение функции не определено», и если он возникает в качестве одного из аргументов, значение любой функции тоже становится $\perp$.

Добавим еще несколько примеров (звёздочка обозначает «было на практике»):

3. * $x \dot{-} y$ (усечённое вычитание): $\max(0, x - y)$.
4. * $S \subset \mathbb{N}^k$ называется примитивно рекурсивным, если:
   (a) χ_s (характеристическая функция f) является примитивно рекурсивной.
   (b) $\exists$ примитивно рекурсивная $f(x_1, \dots, x_k)$ такая, что $S = \{x \in \mathbb{N}^k \mid f(x) = 0\}$

   Два этих определения эквивалентны. Разница между ними в том, что характеристическая функция принимает всего два значения, а вот во втором пункте различных значений функции может быть произвольно много.
5. * Объединение, пересечение, дополнение примитивно рекурсивных множеств примитивно рекурсивно.
6. Примитивно рекурсивный предикат — это частный случай примитивно рекурсивной функции (принимающей только значения 0 или 1). Предикат примитивно рекурсивен тогда и только тогда, когда множество его единиц примитивно рекурсивно. Из предыдущего пункта следует, что если P, Q примитивно рекурсивны, то $P \wedge Q$, $P \vee Q$, $\neg P$ тоже примитивно рекурсивны
7. * Предикат $x = 0$
8. * Предикат $x \geqslant y$
9. * Предикат $x = y$
10. Условный оператор:

$$f(x) = \texttt{if } r(x) \texttt{ then } g(x) \texttt{ else } h(x)$$

    где $r(x)$ — примитивно рекурсивный предикат, а $g(x)$ и $h(x)$ — примитивно рекурсивные функции.

    ▶
    $$f(x) = (1 \dot{-} r(x))h(x) + r(x)g(x)$$
    Если $r(x) = 0$ (неверен), то зануляется первое слагаемое, а второе остаётся. Если $r(x) = 1$, то наоборот. ◀
11. $x \bmod n$.

    ▶ Для начала определим функцию $h(x, n)$ которая прибавляет единицу по модулю, если $x < n$:
    $$h(x) = \texttt{if } x + 1 < n \texttt{ then } x + 1 \texttt{ else } 0$$
    Теперь определяем модуль рекурсивно примитивно:
    $$0 \bmod n = 0$$
    $$(x + 1) \bmod n = h(x \mod n, n)$$
    ◀

12. * Если $R(x,y)$ — примитивно рекурсивный предикат, то следующие два предиката тоже примитивно рекурсивны («ограниченные кванторы»):

    (a) $S(x,z) = \exists y \leqslant z\colon R(x,y)$

    (b) $T(x,z) = \forall y \leqslant z\colon R(x,y)$

13. Пусть Γ_f (график функции f, множество пар) — примитивно рекурсивное множество, причём $f(x) \leqslant g(x)$ (где $g(x)$ примитивно рекурсивна). Тогда $f(x)$ тоже примитивно рекурсивна. Тут нам неважно, что x — это один аргумент, это может быть целый вектор.

    ▶ Пусть $\Gamma(x,y)$ — характеристическая для Γ_f. Тогда можно записать:

    $$f(x) = \sum_{y=0}^{\infty} \Gamma(x,y) \cdot y$$

    В этой бесконечной сумме у нас ровно одно ненулевое слагаемое. К сожалению, бесконечных сумм у нас нет, но мы знаем, что можно суммировать только до $g(x)$:

    $$f(x) = \sum_{y=0}^{g(x)} \Gamma(x,y) \cdot y$$

    Осталось показать, что суммирование $\sum_{y=0}^{n} h(y)$ примитивно рекурсивно (если $h(y)$ примитивно рекурсивно). Потом подставим нужную верхнюю границу.
    Доказываем операцией примитивной рекурсии:
    - $\sum_{y=0}^{0} h(y) = h(0)$
    - $\sum_{y=0}^{n+1} h(y) = \left(\sum_{y=0}^{n}\right) + h(y+1)$

    ◀

14. Ограниченный оператор минимизации примитивно рекурсивен:

    $$f(x) = \begin{cases} \min_{y \leqslant g(x)}\{y\colon R(x,y)\}, \\ g(x)+1, \text{ если такого } y \text{ нет} \end{cases}$$

    Здесь $g(x)$ примитивно рекурсивна.

    ▶ Воспользуемся предыдущим пунктом. По определению $f(x) \leqslant g(x)+1$. Осталось понять, что график примитивно рекурсивен. Если игнорировать тот факт, что y может не найтись, то можно написать вот такую характеристическую функцию для графика:

    $$h(x,y) = R(x,y) \wedge (\underbrace{\forall y' < y\colon \neg R(x,y')}_{\text{примитивно рекурсивно по пункту 12}})$$

    Это примитивно рекурсивная функция. Можно даже добавить в конец $\wedge(y \leqslant g(x))$, но тогда у нас в некоторых точках x графика просто не будет. Лечится условным оператором:

    $$\Gamma_f(x,y) = \texttt{if } (y \leqslant g(x)) \texttt{ then } h(x,y) \texttt{ else } (y = g(x)+1) \wedge (\forall y' < y\colon \neg R(x,y'))$$

    Альтернативно можно дописать к h условие $y \leqslant g(x)$ и логическое «или» результата с веткой `else`. По смыслу то же самое, но без оператора `if`:

    $$\Gamma_f(x,y) = (h(x,y) \wedge (y \leqslant g(x))) \vee \vee ((y = g(x)+1) \wedge (\forall y' < y\colon \neg R(x,y')))$$

    ◀

15. Кодирование пар: $[x, y] = (x + y)^2 + y$. Для разных пар это разные значения (потому что $(x + y)^2 \leqslant (x + y)^2 + y < (x + y + 1)^2$) Это не взаимооднозначное кодирование, но по паре мы умеем однозначно восстанавливать x и y:

$$p_1(n) = \min\{x \leqslant n \colon \exists y \leqslant n : [x, y] = n\}$$

Аналогично восстанавливается $p_2(n)$. Теперь можно строить даже тройки, четвёрки и так далее (например, объединяя пары).

Теорема 1.15.1. Если g — примитивно рекурсивная функция и $f(x)$ вычисляется на машине Тьюринга за $\leqslant g(x)$ шагов, то f тоже примитивно рекурсивна.

Замечание 1.15.2. Альтернативно на паскале: **TODO**

▶ Был рассказан некий план решения, «который вы можете самостоятельно дополнить деталями». Пусть $|\Sigma| = m$ — число символов в алфавите машины Тьюринга. Мы уже кодировали конфигурацию машины Тьюринга при помощи натуральных чисел: стэк элементов левее головки l, стэк элементов правее головки r, символ c под головкой, состояние q. Стэки мы храним, как числа в системе счисления с основанием m: $a_0 + ma_1 + m^2a_2 + \ldots$ (если a_0 — элемент на вершине).

Дальше нужна функция $\texttt{Init}(x)$, возвращающая изначальное состояние машины (закодированную четвёрку), то есть ленту с x, разложенным в двоичную систему. Также нужна функция $\texttt{Result}(x)$, которая считывает ответ из конфигурации и возвращает. Это можно выразить через имеющиеся примитивно рекурсивные функции.

Осталось реализовать функцию работы машины $M(k, t)$ — возвращает конфигурацию, которая получится из k через t шагов. Она определяется примитивно рекурсивно: в $t = 0$ это просто подстановка, а для больших нужно написать здоровенный `if` по текущему состоянию. Например, операция «положить в стэк» — это домножение на m и сложние, «достать из стэка» — взятие остатка, сдвиг головки — перекладывание между стэками и текущим символом. Это пишется честно, но довольно неприятно.

Теперь просто запускаем $\texttt{Result}(M(\texttt{Init}(x), g(x)))$ и получаем результат. Будем считать, что если машина Тьюринга завершила работу, то конфигурация просто перестаёт меняться, тогда нам не надо искать момент остановки ограниченной минимизацией. ◀

1.15.1. Частично-рекурсивные функции

Def 1.15.4. Пусть $f \colon M \to \mathbb{N}$, где $M \subseteq \mathbb{N}^k$. Тогда частично-рекурсивными f считаем следующие:

1. Примитивно рекурсивные функции
2. Все функции, которые можно получить из частично рекурсивных следующими операциями:
   (a) Подстановка
   (b) Примитивная рекурсия
   (c) Оператор минимизации: пусть есть частично рекурсивная функция $g(x_1, \ldots, x_k)$, тогда частично рекурсивна

$$f(x_1, \ldots, x_k) = \min\{y \colon g(x_1, \ldots, x_k, y) = 0\}$$

Обозначается так:

$$f(x) = \mu y \colon g(x, y) = 0$$

Иногда f может получиться где-то неопределена, тогда мы возвращаем неопределённость $\perp$.

Если один из параметров функции получается $\perp$, то она сразу возвращает $\perp$.

Замечание 1.15.3. Существенное отличие от примитивно рекурсивных функций — минимазиция без ограничений.

Теорема 1.15.2. Частично рекурсивная функция вычислима на машине Тьюринга.

▶ Просто берём и по определению и тезису Чёрча вычисляем. Оператор минимизации — это цикл `while`, который, возможно, зацикливается в случае неопределённости. ◀

Теорема 1.15.3. Функция f, вычислимая на машине Тьюринга, является частично рекурсивной.

▶ Рассмотрим предикат $S(x, y, t)$ — машина Тьюринга на входе x останавливается за $\leqslant t$ шагов и печатает y. По теореме 1.15.1 понятно, что S примитивно рекурсивен (упражнение, просто делаем то же самое). Теперь минимизируем по z (закодированная пара $z = [y, t]$):

$$f(x) = p_1(\mu z\colon \neg S(x, p_1(z), p_2(z)))$$

Напоминание: функция p_1 возвращает первый элемент пары. На самом деле нам неважно, что тут именно минимум — ответ от этого не поменяется. Отрицание нужно, потому что в определении минимизации мы искали корень, а не единицу. ◀

Следствие 1.15.3.1 (Нормальная форма Клини). Если f — частично рекурсивная функция, то существуют такие примитивно рекурсивные a и b такие, что:

$$f(x) = a(\mu z\colon b(x, z) = 0)$$

▶ Просто вычисляем f на машине Тьюринга, а затем строим по доказательству теоремы требуемый вид. ◀

Утверждение 1.15.1. Перечислимое множество — проекция примитивно рекурсивного множества пар. Напоминание: множество пар примитивно рекурсивно, если оно имеет вид $\{(\alpha, \beta) \mid f(\alpha, \beta) = 0\}$, где f примитивно рекурсивна, а проекция — это просто множество первых элементов пар.

▶ A перечислимо $\Rightarrow A$ — область определения некоторой вычислимой функции f (напоминание: берём полуразрешающий алгоритм, из него получаем функцию f). Представим f в нормальной форме Клини (вычислимая $\iff$ частично рекурсивная):

$$f(x) = a(\mu z\colon b(x, z) = 0)$$

Здесь a и b примитивно рекурсивны. Теперь рассмотрим множество $B = \{(x, z) \mid b(x, z) = 0\}$. Оно, очевидно, примитивно рекурсивно по построение. А проекция первого элемента этого множества — в точности те x, для которых существует некоторое z такое, что $b(x, z) = 0$, то есть в точности область определения $f(x)$. ◀

1.15.2. Скорость роста примитивно рекурсивных

Хотим выяснить, существует ли всюду определённая вычислимая функция, не являющаяся примитивно рекурсивной.

Def 1.15.5. n-кратное применение функции f к x будем обозначать $f^{[n]}(x)$:

$$f^{[n]}(x) = \underbrace{f(f(\ldots(f(}_{n \text{ вызовов}} x))\ldots))$$

Def 1.15.6. Функция Аккермана $a_i(x)$ от двух параметров (просто с индексами писать удобнее) при $i, x \geqslant 0$:

$$a_0(x) = x + 1$$
$$a_{i+1}(x) = a_i^{[x+2]}(x)$$

Свойства:

1. $a_i(x) > x$ (индукция по i)
2. Для всех i: $a_i(x)$ монотонно возрастает по x, т.е. $a_i(x) < a_i(x+1)$.

   ▶ Индукция по i:

   База: $i = 0$ — $a_i(x) = x + 1$, очевидно.

   Переход: $i \to i+1$ — доказали для a_i, показываем для a_{i+1}. Пусть $y > x \geqslant 0$ Тогда имеем:
   $$a_{i+1}(y) = a_i^{[y+2]}(y) > a_i^{[y+2]}(x) > a_i^{[x+2]}(x) = a_{i+1}(x)$$

   ◀
3. Для всех x: $a_i(x)$ монотонно возрастает по i.
   $$a_{i+1}(x) = a_i^{[x+2]}(x) > a_i(x)$$
4. $a_i(a_i(x)) \leqslant a_{i+1}(x)$, так как при $x = 0$ имеем в точности равенство, а при $x > 0$:
   $$a_{i+1}(x) = a_i^{[x+2]}(x) = a_i(a_i(a_i^{[x]}(x))) > a_i(a_i(x))$$

Теорема 1.15.4. Если $f(x_1, \dots, x_n)$ — примитивно рекурсивная функция, то для некоторого N имеем:
$$\forall x_1, \dots, x_n\colon f(x_1, \dots, x_n) \leqslant a_N(\max\{x_1, \dots, x_n\})$$

▶ Индукция по построению примитивно рекурсивных функций. Чуть формальнее: каждая примитивно рекурсивная функция получается за несколько операций из базисных, индукция по числу этих операций.

1. Базисные функции — подходит $N = 0$:
   (a) Для 0 очевидно
   (b) $s(x) = x + 1 \leqslant a_0(x)$
   (c) $\pi_n^k(x_1, \dots, x_n) = x_k \leqslant \max\{x_1, \dots, x_n\} \leqslant a_0(\max\{x_1, \dots, x_n\})$
2. Подстановка: $g(x_1, \dots, x_n) = f(h_1(x_1, \dots, x_n), h_2(x_1, \dots, x_n), \dots, h_k(x_1, \dots, x_n))$.
   Положим $\vec{x} = (x_1, \dots, x_n)$, а $\max\{x_1, \dots, x_n\} = x_{\max}$. Тогда знаем, что для некоторых N_0 (оценка на функцию f) и $N_1, \dots, N_k$ (оценки на функции h_i):
   $$\begin{aligned} g(\vec{x}) = f(\dots) &\leqslant a_{N_0}(\max\{h_1(\vec{x}), h_2(\vec{x}), \dots, h_k(\vec{x})\}) \\ h_1(\vec{x}) &\leqslant a_{N_1}(x_{\max}) \\ &\vdots \\ h_k(\vec{x}) &\leqslant a_{N_k}(x_{\max}) \end{aligned}$$

Тогда положим $N' = \max\{N_0, N_1, \dots, N_k\}$:

$$\begin{aligned} g(\vec{x}) &\leqslant a_{N'}(\max\{h_1(\vec{x}), h_2(\vec{x}), \dots, h_k(\vec{x})\}) \\ h_1(\vec{x}) &\leqslant a_{N'}(x_{\max}) \\ &\vdots \\ h_k(\vec{x}) &\leqslant a_{N'}(x_{\max}) \end{aligned}$$

Отсюда легко видно, что:

$$g(\vec{x}) \leqslant a_{N'}(\max\{h_1(\vec{x}), h_2(\vec{x}), \dots, h_k(\vec{x})\}) \leqslant a_{N'}(a_{N'}(x_{\max})) \leqslant a_{N'+1}(x_{\max})$$

Таким образом, нашли искомое число: $N = N' + 1$.

3. Примитивная рекурсия (аналогично обозначили $\vec{x}$):

$$\begin{aligned} g(\vec{x}, 0) &= f(\vec{x}) \\ g(\vec{x}, y+1) &= h(\vec{x}, y, g(\vec{x}, y)) \end{aligned}$$

Возьмём общую оценку N' на f и h:

$$\begin{aligned} f(\vec{x}) &\leqslant a_{N'}(x_{\max}) = a_{N'}(\max\{x_{\max}, 0\}) \\ h(\vec{x}, y) &\leqslant a_{N'}(\max\{x_{\max}, y\}) \end{aligned}$$

Индукцией по i покажем, что $g(\vec{x}, i+1) \leqslant a_{N'}^{[i+2]}(\max\{x_{\max}, i\})$

База $i = 0$**:**

$$\begin{aligned} g(\vec{x}, 0+1) = h(\vec{x}, 0, g(\vec{x}, 0)) &\leqslant a_{N'}(\max\{x_{\max}, 0, g(\vec{x}, 0)\}) \leqslant \\ &\leqslant a_{N'}(\max\{x_{\max}, 0, a_{N'}(x_{\max})\}) \leqslant a_{N'}^{[2]}(\max\{x_{\max}, 0, x_{\max}\}) \end{aligned}$$

Переход $(i-1) \to i$**:** Предполагаем, что $i \geqslant 1$:

$$\begin{aligned} g(\vec{x}, i+1) = h(\vec{x}, i, g(\vec{x}, i)) &\leqslant a_{N'}(\max\{x_{\max}, i, g(\vec{x}, i)\}) \leqslant \\ &\leqslant a_{N'}(\max\{x_{\max}, i, a_{N'}^{[i+1]}(\max\{x_{\max}, i-1\})\}) \leqslant \\ &\leqslant a_{N'}(a_{N'}^{[i+1]}(\max\{x_{\max}, i\})) = (a_{N'}^{[i+2]}(\max\{x_{\max}, i\})) \end{aligned}$$

Теперь мы имеем требуемую оценку на $N = N' + 1$:

$$\begin{aligned} g(\vec{x}, 0) &\leqslant a_{N'}(\max\{x_{\max}, 0\}) \leqslant a_N(\max\{x_{\max}, 0\}) \\ g(\vec{x}, i+1) &\leqslant a_{N'}^{[i+2]}(\max\{x_{\max}, i\}) \\ &\leqslant a_{N'}^{[2+\max\{x_{\max}, i\}]}(\max\{x_{\max}, i\}) = a_N(\max\{x_{\max}, i\}) \leqslant \\ &\leqslant a_N(\max\{x_{\max}, i+1\}) \end{aligned}$$

◀

Следствие 1.15.4.1. Фукнция $b(n) = a_n(n)$ (от одного аргумента n) не является примитивно рекурсивной.

▶ Пусть является. Тогда существует N такое, что $b(n) \leqslant a_N(n)$ для всех n. Рассмотрим случай $n = N + 1$:

$$\begin{aligned} b(n) &\leqslant a_N(n) \\ b(n) = b(N+1) = a_{N+1}(N+1) &> a_N(N+1) = a_N(n) \end{aligned}$$

Противоречие. ◀

Следствие 1.15.4.2. Функция Аккермана от двух аргументов не является примитивно рекурсивной.

▶ Если $a(i, x)$ является примитивно рекурсивной, то $b(n) = a(\pi_1^1(n), \pi_1^1(n))$ тоже является, что не так. ◀

Упражнение 1.15.1. При фиксированном k функция $a_k(x)$, тем не менее, является примитивно рекурсивной.

Глава 2

Исчисление предикатов

2.1. Элиминация кванторов

2.1.1. Мотивация

Давайте вспомним предикатные формулы. Например, такие: носитель интерпретаций — $\mathbb{Z}$, $\mathcal{P} = \{=\}$, $\mathcal{F} = \{+\}$.

Утверждение 2.1.1. Предикат $x < y$ невыразим.

▶ Вспомним *метод автоморфизмов*: мы предъявляем некоторый автоморфизм $\mathbb{Z}$ (биекцию между элементами $\mathbb{Z}$ и $\mathbb{Z}$), сохраняющий все операции (и предикатные, и функциональные) и показываем, что искомый предикат он не сохранит.

В самом деле: возьмём преобразование $x \leftrightarrow (-x)$. Оно является автоморфизмом:

- $x \leftrightarrow (-x)$ — биекция $\mathbb{Z} \leftrightarrow \mathbb{Z}$
- $(x = y) \iff (-x = -y)$
- $-(x + y) = (-x) + (-y)$

При этом легко предъявить два элемента $\mathbb{Z}$ (при этом выразимость этих элементов неважна, важно лишь их существования): 0 и 1, для которых предикат $x < y$ меняется от автоморфизма. Следовательно, предикат $x < y$ невыразим. ◀

Замечание 2.1.1. Для применения метода автоморфизмов важно предъявлять именно автоморфизм, т.е. именно биекцию, причём сохраняющую *все* операции.

Теперь изменим функциональную операцию: возьмём $S(x) = x + 1$ вместо $+$. Оказывается, что предикат $x < y$ всё еще невыразим, однако множество допустимых автоморфизмов сузилось: это только сдвиги вида $x \leftrightarrow x + C$, сохраняющие в том числе отношение $x < y$. Таким образом, метод автоморфизмов тут неприменим.

Более того, если рассмотреть предикатные формулы над $(\mathcal{N}, =, S)$, то автоморфизмов вообще не окажется (кроме тривиального). Соответственно, доказать хоть что-то про невыразимость методом автоморфизмов не получится. Однако это вовсе не означает, что все предикаты выразимы.

Дальше мы покажем более мощный метод, позволяющий, среди прочего, доказывать невыразимость.

2.1.2. Над целыми числами

Теорема 2.1.1. В интерпретации $(\mathbb{Z}, =, S)$ допустима *элиминация кванторов*, то есть любой выразимый предикат можно выразить без кванторов.

Следствие 2.1.1.1. Предикат $x < y$ невыразим.

▶ От противного: пусть $x < y$ выразим и, следовательно, представляется в бескванторной форме. Заметим, что все атомарные формулы имеют вид $S(S(\dots(x)\dots)) = S(\dots(y)\dots)$, то есть задают условия вида $x = y + k$, где $k \in \mathbb{Z}$ — некоторая фиксированная константа. Тогда выбрав из этих констант максимальную по абсолютному значению — K, можно взять любое число x и заметить, что если мы возьмём y, достаточно сильно отличающийся от x (хотя бы на $K+1$ по модулю разности), то все атомарные формулы будут одинаково ложны независимо от выбора y.

В частности, наш предикат не сможет отличить y, сильно больший x от y, сильно меньшего x. Получили противоречие. ◀

Теперь докажем теорему 2.1.1:

▶ Сначала возьмём выразимый предикат P и заменим все входящие в него кванторы всеобщности на кванторы существования:

$$(\forall x\colon A) \iff (\neg(\exists x\colon \neg A))$$

Теперь запустим индукцию по числу кванторов k в формуле. База для $k = 0$ очевидно: формула уже бескванторная.

Переход: рассмотрим самый «внутренний» квантор в формуле, т.е. выражение следующего вида:

$$\exists x\colon \varphi(x_1, \dots, x_n, x)$$

Тут φ — бескванторная. Если мы сможем выразить это выражение без кванторов, то переход будет доказан: мы уменьшим число кванторов и воспользуемся индукционным предположением.

Теперь рассмотрим все автомарные формулы, встречающиеся в φ. Уже знакомыми нам рассуждениями легко показать, что они все эквивалентны уравнениям вида $\alpha = \beta + k$, где k — какие-то константы из $\mathbb{Z}$, а α и β — переменные.

- Рассмотрим атомарные формулы вида $x = x + k$. Очевидно, что при $k = 0$ их можно заменить на тождественную истину, а при $k \neq 0$ — на тождественную ложь.
  Замечание 2.1.2. Без тождественной истины предикат без аргументов $\exists x\colon x = x$ в бескванторной форме не выразить. Если же сказать, что мы выражаем только предикаты с ненулевым числом аргументов, то легко выразить тождественную истину и ложь.
- Значение атомарных формул вида $x_i = x_j + k$ не зависит от x.
- Рассмотрим атомарные формулы вида $x = x_i + k$, назовём их $t_1, t_2, \dots, t_s$. Так как их конечно, то существует некоторый x, при котором ни одна из них не будет выполнена. Построим формулу ψ: это φ, в которой все t_i заменены на тождественную ложь.
  Теперь рассмотрим все возможные значения для x, которые хотя бы одну формулу из t_i превратят в истину: $\alpha_1, \alpha_2, \dots, \alpha_s$. Их тоже конечно, они зависят только от структуры t_i. Тогда можно переписать исходную формулу в бескванторном виде:

$$(\exists x\colon \varphi(x_1, \dots, x_n, x)) \iff (\bigvee_{i=1}^{s} \varphi(x_1, \dots, x_n, \alpha_i)) \vee \psi(x_1, \dots, x_n)$$

  В самом деле: нам сами значения x неважны, а важно лишь то, какие из t_i обращаются в истину. Все возможные варианты мы покрыли.
  Замечание 2.1.3. Если $\alpha \geqslant 0$, то его можно записать в виде $S(S(\dots S(0)\dots))$. А если $\alpha < 0$, то можно временно ввести функцию $P(x) = x - 1$, записать его в виде $P(P(\dots P(0)\dots))$, а потом, когда уже получим атомарные формулы с P и S, добавить с обеих сторон $S()$, пока не избавимся от этой свежедобавленной функции.

◀

Замечание 2.1.4. Мы не только доказали теорему, но и предъявили явный алгоритм. Основная идея была в том, что сами значения x в кванторах нам неинтересны, нам интересно лишь то, какие атомарные формулы в каких комбинацию могут выполняться, а так как атомарных формул конечно, мы смогли предъявить конечное число «интересных» x.

2.1.3. Над вещественными и рациональными

Теорема 2.1.2. В $(\mathbb{Q}, =, >, +, \text{рац. константы})$ допустима элиминация кванторов.

▶ 1. Избавляемся от квантора $\forall$

2. Применяем индукцию и в переходе берём самый вложенный $\exists$:

$$\exists x \colon \varphi(x_1, \dots, x_n, x)$$

3. Избавимся от отрицаний в φ, «пронеся» их внутрь к атомарным формулам по правилам де Моргана, а затем убрав отрицания от атомарных формул:
   - $\neg(x > y) \iff (y > x) \vee (y = x)$
   - $\neg(x = y) \iff (x > y) \vee (y > x)$
4. Приводим формулу в ДНФ, причём так, чтобы новых отрицаний не образовалось. Это можно сделать при помощи алгоритма с раскрытием скобок, используя формулу: $a \wedge (b \vee c) \iff (a \wedge b) \vee (a \wedge c)$. Теперь преобразуем:

$$\begin{gathered}\exists x \colon (C_1 \vee C_2 \vee \dots \vee C_k) \\ \Updownarrow (\exists x \colon C_1) \vee (\exists x \colon C_2) \vee \dots \vee (\exists x \colon C_k)\end{gathered}$$

   Количество кванторов мы, конечно, увеличили, но сейчас избавимся от каждого в отдельности — случаи стали приятнее.
5. Теперь разберёмся с элиминацией кванторов в выражении $\exists x \colon C_i$. Так как мы получили C_i из ДНФ без отрицаний, C_i представляет собой конъюнкцию атомарных формул без отрицаний. Каждая атомарная формула — это линейное уравнение или неравенство над $x_1, \dots, x_n, x$. В каждой атомарной формуле, где есть x, можно перенести x влево, всё остальное — вправо, и нормализовать коэффициент при x при помощи деления в $\mathbb{Q}$ (возможно, со сменой знака неравенства).
   Таким образом, C_i — это просто система уравнений и неравенств. Условия, не содержащие x, можно оставить, как есть. Разберёмся с оставшимися, есть три случая:
   - Если есть условие вида $x = \dots$, то можно взять любое и подставить во всём выражении φ вместо x соответствующее выражение, получим выражение без x, с которого можно безболезненно снять квантор $\exists x$.
   - Если у нас неравенства на x только с одной стороны (например, $x < x_1$ и $x < x_3 - x_2$), то все эти неравенства можно просто убрать — очевидно, существует достаточно маленький/большой x.
   - Остался единственный случай: у нас есть неравенства на x с двух сторон, например:

$$\begin{cases} x < t_1, \\ x < t_2, \\ \vdots \\ x < t_s, \\ x > z_1, \\ x > z_2, \\ \vdots \\ x > z_r, \end{cases}$$

Тогда существование нужного x есть в точности система $\bigwedge_{i=1}^{r}\bigwedge_{j=1}^{s} z_i < t_j$.

◀

Замечание 2.1.5. В $(\mathbb{R}, =, >, +,$ вещ. константы) тоже допустима элиминация кванторов — мы нигде не пользовались рациональностью $\mathbb{Q}$. Мы пользовались только отсутствием наибольших и наименьших элементов, плотностью и возможностью делить.

Следствие 2.1.2.1. Любая *замкнутая* формула из $(=, >, +,$ рац. константы) одинаково верна и в $\mathbb{Q}$, и в $\mathbb{R}$. Например: $\forall x\colon \exists y\colon x + x = y$.

▶ Применим наш алгоритм элиминации к формуле над $\mathbb{R}$ и к той же формуле над $\mathbb{Q}$, получим одинаковый результат (так как алгоритм действует одинаково). Так как формула была замкнутой, мы получим либо тождественную ложь, либо тождественную истину. ◀

Замечание 2.1.6. Умножение добавлять в сигнатуру нельзя. Например: $\exists x\colon x \cdot x = 2$ верна в $\mathbb{R}$, но не в $\mathbb{Q}$.

2.1.4. Пример задачи

Теорема 2.1.3. Пусть имеется единичный квадрат, разрезанный на несколько квадратов со сторонами $r_1, r_2, \dots, r_k$. Тогда стороны этих квадратов рациональны.

▶ Напишем формулу с параметрами $r_1, \dots, r_k$, которая проверяет существование разбиения: существуют такие координаты углов $(x_1, y_1), \dots, (x_k, y_k)$, что никакие два квадратика не пересекаются (то есть не пересекаются либо их проекции по x, либо их проекции по y), все квадратики лежат внутри единичного, а также не существует никакой точки, лежащей строго снаружи всех квадратиков разбиения (на лекции говорили про суммарную площадь квадратиков, но так нельзя: у нас нет операции умножения). Каждое из этих условий — это какие-то неравенства/уравнения от координат и размеров квадратиков.

В этой формуле можно элиминировать кванторы, избавиться от отрицания и привести в ДНФ. Рассмотрим каждый конъюнкт по отдельности, покажем, что все его решения рациональны. Конъюнкт эквивалентен системе некоторых неравенств и уравнений на r_i. Системы без решений сразу рассматривать не будем — они неинтересны.

Возьмём систему, у которой есть некоторое решение r. Сначала посмотрим отдельно на уравнения, входящие в эту систему. Если решение подсистемы из уравнений единственно, то оно рационально (так как все коэффициенты рациональны и можно применить формулу Крамера). Если же решений подсистемы уравнений бесконечно много, то имеется целая прямая этих решений, причём проходящая через r. Неравенства на этой прямой высекут некоторый открытый отрезок. Очевидно, он непуст, так как r в нём точно лежит.

Теперь заметим, что целого отрезка решений системы из уравнений и неравенств у нас быть не может. В самом деле: пусть у нас есть отрезок $[r, r + h]$, то есть любой набор квадратов вида $r_1 + th_1, \dots, r_k + th_k$ замощает единичный квадрат (при $t \in [0, 1]$). Теперь из геометрических соображений получим противоречие: суммарная площадь квадратиков, замощающих единичный квадрат, должна быть равна единице. Посчитаем суммарную площаль квадратиков, получим некоторый многочлен от t степени 2, причём старший коэффициент (равный $h_1^2 + h_2^2 + \dots + h_k^2$) нулю не равен. Стало быть, этот многочлен не может быть равен единице на отрезке $[0, 1]$, противоречие. ◀

2.1.5. В элементарной теории вещественных чисел

Мотивация

Утверждение 2.1.2. Элиминация кванторов возможна в следующей арифметике над $\mathbb{R}$: $\mathcal{P} = \{=, >\}$, $\mathcal{F} = \{+, \cdot, 0, 1\}$.

Замечание 2.1.7. $x > 0$ не выразить бескванторной формулой в $(\mathbb{R}, =, +, x, 0, 1)$

Замечание 2.1.8. Если разрешать любые формулы, то 0 и 1 не нужны, да и $>$ тоже:

- $x = 0 \iff \forall y\colon x + y = y$
- $x = 1 \iff \forall y\colon x \cdot y = y$
- $x > y \iff \exists z\colon x = y + z^2$.

Как вообще выглядят атомарные формулы? Всего два вида: $P(x_1, \dots, x_n) = 0$ или $P(x_1, \dots, x_n) > 0$, где P — многочлен с целыми коэффициентами. Обращаем внимание, что если добавить рациональные константы, то ничего бы не изменилось — многочлен всегда можно домножить на общий знаменатель и получить многочлен с целыми коэффициентами.

Далее будет рассказан алгоритм, который не слишком эффективен (и вообще плохо оценивается по скорости). Лучший результат в этой области оценивается как двойная экспонента (2^{2^n}), при этом есть нижняя оценка того же порядка.

Что вообще нам даст элиминация кванторов алгоритмом? Можно любую формулу в элементарной теории вещественных чисел проверить: свести к бескванторной форме, а потом просто подставить.

Пример 2.1.1. Любую задачу школьной геометрии, где используются только прямые, окружности (и их куски), которые касаются и пересекаются, можно записать в координатах, получить какой-то набор условий в арифметике, а потом алгоритмически доказать или проверить.

Пример 2.1.2. В школе учат элиминировать кванторы в выражении вида $\exists x\colon (x^2 + px + q = 0)$. Это просто условие на существование решения квадратного уравнения, можно записать бескванторно: $p^2 - 4q \geqslant 0$ (условие на дискриминант). Для уравнений бóльшей степени в школе обычно не изучают, однако из того, что мы сейчас докажем, следует, что для любых уравнений и неравенств подобные эквивалентности существуют.

Основные определения

Будет рассказан так называемый алгоритм Тарского (Тарского-Зайденберга). Изначальная версия была сложной, но есть более простая:

Теорема 2.1.4 (Тарского-Зайденберга)**.** В $(\mathbb{R}, =, >, +, \cdot, 0, 1)$ допустима элиминация кванторов.

Def 2.1.1. Алгебраическое множество — это решение системы уравнений из многочленов.

Def 2.1.2. Полуалгебраическое множество в $\mathbb{R}^n$ — это объединение конечного числа множеств решений систем полиномиальных уравнений и неравенств.

Замечание 2.1.9. Например:

$$\left[\begin{array}{l} \begin{cases} xy + y^5 - 20 = 0 \\ z^2 - 3 = 0 \\ x + y \geqslant 0 \end{cases} \\ \begin{cases} xy + y^5 - 20 = 0 \\ z^2 = 3 \\ z^{100} \leqslant x \end{cases} \end{array}\right.$$

Замечание 2.1.10. Чуть более строго: полуалгебраическое множество — это множество таких $\vec{x}$, для которых выполняется некоторое условие следующего вида:

$$\bigvee_i \bigwedge_j P_{ij}(\vec{x}) \gtreqless 0$$

Здесь P_{ij} — многочлены от многих переменных с целыми коэффициентами.

Утверждение 2.1.3. Объединение полуалгебраических множеств тоже полуалгебраическое.

▶ Очевидно: просто объединяем две дизъюнкции. ◀

Упражнение 2.1.1. Пересечение полуалгебраических множеств полуалгебраично.

▶ Подсказка: надо взять конъюнкцию условий, а дальше раскрыть скобки по дистрибутивности. Условно, привести в ДНФ. ◀

Утверждение 2.1.4. Любое полуалгебраическое множество задаётся некоторой бескванторной формулой.

▶ Это очевидно: формулу без кванторов (но с многочленами) мы можем написать по определению, а многочлены — вполне себе бескванторные формулы. ◀

Утверждение 2.1.5. Любая бескванторная формула задаёт полуалгебраическое множество.

▶ Сначала избавимся от отрицания:

1. $\neg(a = b) \longrightarrow (a < b) \vee (a > b)$
2. $\neg(a > b) \longrightarrow (a = b) \vee (a < b)$

Приводим в ДНФ (алгоритмом раскрытия скобок, при котором отрицаний не возникает). Получаем в точности определение полуалгебраического множества. ◀

Проеция полуалгебраического

Утверждение 2.1.6. Проекция полуалгебраического множества из $\mathbb{R}^{n+1}$ вдоль оси координат является полуалгебраическим множеством в $\mathbb{R}^n$.

Пример 2.1.3. Пусть было полуалгебраическое множество A в $\mathbb{R}^3$, задающее «шапку» сферы:

$$(x^2 + y^2 + z^2 = 1) \wedge \left(z > \frac{1}{2}\right)$$

Тогда мы можем спроецировать A вдоль оси OZ, получить некоторое множество B на плоскости OXY. Нетрудно понять, что это на самом деле круг, то есть тоже полуалгебраическое множество:

$$x^2 + y^2 < \frac{3}{4}$$

Замечание 2.1.11. Утверждение 2.1.6 эквивалентно теореме Тарского-Зайденберга.

▶ Если теорема верна, то утверждение очевидно: это просто навешивание снаружи квантора существования по оси.

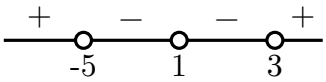
В другую сторону: можно заменить кванторы $\forall$ на $\exists$ и запустить индукцию по числу кванторов (как обычно). Пусть была формула с кванторами существования, взяли самый внутренний: $\exists x_i \colon F$, причём F бескванторна. Тогда F полуалгебраическое, а по утверждению его проекция, т.е. множество $\exists x_i \colon F$ — полуалгебраическое, т.е. можно переписать этот кусок формулы без кванторов, что и требовалось доказать для перехода в индукции. ◀

Основная идея доказательства: метод интервалов из школьной программы.

Пример 2.1.4. В школе рассказывают метод интервалов: найти, где какой знак имеет следующий многочлен:

$$(x-1)^2(x-3)(x+5) > 0$$

На бесконечности многочлен имеет знак плюс, в корнях знак как-то меняется, Давайте нарисуем корни на числовой прямой. Свели бесконечное число разных x к лишь конечному числу случаев:



Теперь давайте избавляться от квантора существования в общем случае. Есть формула:

$$\exists x\colon \varphi(x, x_1, \dots, x_n)$$

Мы считаем, что все атомарные формулы имеют вид $P_i(x, x_1, \dots, x_n) \gtreqless 0$. Будет удобно считать, что у нас на самом деле многочлен от одной переменной x, а вот коэффициенты при степенях — многочлены от $x_1, \dots, x_n$. По сути следующая замена:

$$\mathbb{Z}[x, x_1, \dots, x_n] = \mathbb{Z}[x_1, \dots, x_n][x]$$

Выпишем все многочлены из $\mathbb{Z}[x, x_1, \dots, x_n]$, которые встретились в атомарных формулах: $P_1, \dots, P_k$. Заметим, что если мы подставим $x_1 = a_1, x_2 = a_2, \dots$ ($a_i \in \mathbb{R}$), то у нас все этим многочлены P_i станут многочленами от x.

Диаграммы

Пусть $Q_1, \dots, Q_k$ — многочлены от одной переменной. (нас будут интересовать те, которые получились из P_k, но вообще сейчас на это опираться не будем).

Def 2.1.3. Пусть $\alpha_1, \alpha_2, \dots, \alpha_n$ — это все различные корни ненулевых многочленов, причём в порядке возрастания. Диаграммой для многочленов называется некоторая табличка из k строк и $2n+1$ столбцов. Рисуем про каждый интервал между соседними корнями (и в корнях тоже) нули, плюсы, минусы:

	...	α_1	...	α_2	... ...	α_n	...
Q_1	$\operatorname{sign} Q_1(-\infty)$	$\operatorname{sign} Q_1(\alpha_1)$	$\operatorname{sign} Q_1(\frac{\alpha_1+\alpha_2}{2})$	$\operatorname{sign} Q_1(\alpha_2)$	... ...	$\operatorname{sign} Q_1(\alpha_n)$	$\operatorname{sign} Q_1(+\infty)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Q_k	$\operatorname{sign} Q_k(-\infty)$	$\operatorname{sign} Q_k(\alpha_1)$	$\operatorname{sign} Q_k(\frac{\alpha_1+\alpha_2}{2})$	$\operatorname{sign} Q_k(\alpha_2)$	... ...	$\operatorname{sign} Q_k(\alpha_n)$	$\operatorname{sign} Q_k(+\infty)$

Причём в диаграмму список корней и многочленов *не входят*. Однако порядок многочленов при этом фиксирован, то есть переставлять строки диаграммы нельзя (не говоря уже о столбцах).

Пример 2.1.5. Пусть есть многочлены $x^2 - 1$, x, 0, у них корни $-1, 0, 1$.

	...	-1	...	0	...	1	...
x^2-1	$+$	0	$-$	$-$	$-$	0	$+$
x	$-$	$-$	$-$	0	$+$	$+$	$+$
0	0	0	0	0	0	0	0

Так как ни корни, ни многочлены не входят в диаграмму, для $x^2 - 4$, $2x$, 0 диаграмма получилась бы такая же.

Замечание 2.1.12. Имея диаграмму для многочленов P_i с подставленными $x_1, \dots$ легко определить истинность логической формулы, потому что мы знаем вообще все возможные варианты знаков, которые бывают у многочленов при фиксированных $x_1, \dots$ и произвольном x. Мы просто перебираем конечное число вариантов, подставляем в логическую формулу, вычислили значение формулы.

Утверждение 2.1.7. Множество таких $(a_1, \dots, a_n) \in \mathbb{R}^n$, что при подстановке $x_1 = a_1, \dots, x_n = a_n$ получается некоторая фиксированная диаграмма, является полуалгебраическим.

Замечание 2.1.13. Если докажем, то докажем и утверждение 2.1.6. В самом деле: вспомним про многочлены P_i. При разных подстановках $x_1, \dots, x_n$ у нас получаются разные наборы многочленов $Q_i(x)$, эти наборы дают разные диаграммы. Диаграмм конечно и ограниченно — количество столбцов не больше количества корней у всех многочленов $Q_i(x)$ (то есть не больше суммарной степени по x у P_i), количество строк тоже ограничено (не более, чем у нас P_i), в каждой ячейке диаграммы — одно значение из трёх.

Тогда для бесквантворного выражения проекции можно просто перебрать все диаграммы, из них выбрать те, которые нас устраивают (т.е. есть корректный столбец), для каждой диаграммы узнать, какие подстановки $x_1, \dots, x_n$ нам дадут нужную диаграмму — это что-то полуалгебраическое. Потом просто объединяем все такие полуалгебраические.

Замечание 2.1.14. Другое объяснение: у нас есть $\mathbb{R}^n$ (то, куда проецируем), оно разбилось на какие-то куски, в каждом куске — ровно одна диаграмма. Некоторые диаграммы годятся (т.к. обращают логическую формулу в истину), а некоторые — нет. Ответ, который мы хотим найти — объединение всех годных диаграмм.

Доказательство утверждения 2.1.7

Идея: нам будет удобно *добавлять* лишние многочлены.

Утверждение 2.1.8. Пусть мы доказали, что для конкретных многочленов $P_1, \dots, P_{k+1}$ (лежащих в $\mathbb{Z}[x, x_1, \dots, x_n]$) и любой диаграммы у нас множество соответствующих подстановок переменных $x_1, \dots, x_n$ полуалгебраическое. Тогда, выкинув P_{k+1}, мы тоже получим, что для любой диаграммы множество допустимых подстановок — полуалгебраическое.

▶ Перебрали все диаграммы для $k+1$ многочлена, выкинули строчку для многочлена P_{k+1}, получили какую-то диаграмму для $P_1, \dots, P_k$. Тогда получается, что для каждой диаграммы набора $P_1, \dots, P_k$ искомое множество есть объединение некоторых множеств, соответствующих каким-то диаграммам расширенного набора многочленов (такие множества полуалгебраические), т.е. искомое множество тоже полуалгебраическое. ◀

Теперь вопрос: как мы будем добавлять многочлены и какие, чтобы стало проще? Будем *насыщать* наше множество. Вспомним, что мы можем многочлены P записывать, как многочлены от x с коэффициентами из $\mathbb{Z}[x_1, \dots, x_n]$ и введём операции, которые добавляют что-то в текущее множество многочленов:

1. Взятие старшего члена: был многочлен

$$P = Q_1 x^n + Q_2 x^{n-1} + \dots$$

Добавляем многочлен $Q_1(x_1, \dots, x_n)$, не зависящий от x.

2. Вычёркивание старшего члена: был многочлен

$$P = Q_1 x^n + Q_2 x^{n-1} + \dots + Q_{n+1}$$

добавили многочлен

$$Q_2 x^{n-1} + \dots + Q_{n+1}$$

3. Формальная производная P по переменной x: $\frac{\partial P}{\partial x}$:

$$Q_1 x^n + Q_2 x^{n-1} + \dots$$
$$\downarrow$$
$$(nQ_1)x^{n-1} + ((n-1)Q_2)x^{n-2} + \dots$$

4. Модифицированный остаток от деления P на Q (напоминаем, что $P, Q \in \mathbb{Z}[x_1, \dots, x_n][x]$). Это просто обычный остаток от деления (как многочленов с переменной X), но дополнительно домноженный на старший коэффициент Q в большой степени, чтобы не вылезти из кольца многочленов с *целыми* коэффициентами.

   $$Q = r_1(x_1, \dots, x_n)x^m + \dots$$

   Мы берём $(p \bmod Q) \cdot r_1^C$, где C — константа, зависящая от P и Q. Давайте брать такую минимальную константу, чтобы получился многочлен.

Давайте насытим множество многочленов относительно операций 1–4. То есть берём и применяем операции как угодно, пока получаются новые многочлены.

Утверждение 2.1.9. Этот процесс остановится через конечное число шагов

▶ Доказательство из Шеня. В самом деле: рассмотрим множество всех многочленов, которые мы так хоть когда-то получим (то есть объединение всех шагов). Оно, очевидно, замкнуто относительно операций: у каждой операции лишь конечное число аргументов, и если мы хотим применить операцию к многочленам A и B, то результат точно появится на каком-то конечном шаге.

Осталось показать, что это множество конечно. Заметим, что результат любой операция имеет строго меньшую степень, чем все аргументы операции. Возьмём произвольный ненулевой многочлен X из замыкания. Он получен как результат некоторого применения операций, можно нарисовать такое дерево операций. Так как все исходные аргументы имеют степень не больше M (где M — максимальная степень многочлена среди исходных), то глубина этого дерева не больше M, а у каждой вершины — не больше двух сыновей (по числу аргументов). Таким образом, количество возможных деревьев операций конечно, стало быть, количество многочленов в замыкании тоже ограничено. ◀

Пусть F_0 — это многочлены степени ноль (по x) из насыщенного множества A. То есть это все многочлены вида $P(x_1, \dots, x_n)$, не зависящие от x. Тогда утверждается, что диаграмма для F_0 (которая будет состоять из одного столбца и, понятное дело, соответствующие значения $x_1, \dots, x_n$ являются полуалгебраическим множеством — просто пишем условия на P'шки) позволяет однозначно восстановить диаграмму для исходного A.

Воссстанавливаем диаграмму индукцией по степени. Для степени 0 всё есть по условию, теперь пусть у нас есть диаграмма для многочленов степени $\leqslant k$. Добавляем какой-то многочлен P степени $k+1$.

1. Посмотрели на строчку, соответствующую старшему коэффициенту P (помним, что коэффициенты P — это многочлены). Если в ней везде нули, то надо просто повторить строчку для многочлена P без старшего коэффициента (такой у нас в диаграмме есть). Например, везде нули могли получиться, если у нас особо удачные a_i, зануляющие старший коэффициент P. Если же нули не везде, то надо сделать следующее:

2. Посчитать значение в старых корнях. Взяли старый корень α многочлена Q, поделили P с остатком (модифицированным) на Q, получили $R + Q \cdot (\dots) = P \cdot \beta^C$ (тут β — старший коэффициент P, а C — константа). Тогда $Q \cdot (\dots)$ занулился, знак R у нас есть, надо его, возможно, поменять (так как знак β есть, C есть)

3. Добавить новые корни P (в них значение — ноль). Кратные корни — это корни производной, поэтому они уже есть. Размышление: если у нас в какой-то строчке получилось `+-` или `-+`, то надо добавить между ними корень. Значение корня нам в целом неважно, важен порядок. Утверждение: случаев вроде «`++` и надо добавить чётное число корней» или «`+-`, надо добавить больше одного корня между ними» не бывает, так как по теореме Ролля между любыми двумя корнями есть корень производной.

4. Осталось вычислить промежуточные значения. На плюс бесконечности значение определяется знаком старшего коэффициента. Дальше, когда мы переходим через какой-то корень, в нём либо P не ноль (тогда знак не меняется), либо ноль (тогда надо понять кратность) и знак может как поменяться, так и не поменяться. Кратность P в точке считается методом «взяли много производных, посмотрели сколько из них подряд занулилось».

5. А у старых многочленов новые точки и промежутки просто вложены в старые промежутки и однозначно продлевается.

2.1.6. В комплексных числах

Теорема 2.1.5. В $(\mathbb{C}, =, +, \cdot, 0, 1)$ допустима элиминация кванторов

▶ Атомарные формулы имеют вид $P(x_1, \dots, x_n) = 0$. Назовём множество «хорошим», если оно представляется в виде такого объединения:

$$\bigcup_i \begin{cases} P_{i1} = 0, \\ P_{i2} \neq 0, \\ P_{i3} = 0, \\ P_{i4} \neq 0, \\ \vdots \end{cases}$$

Будем действовать аналогично доказательству теоремы Тарского-Зайденберга. Нам потребуется доказать, что проекция хорошего множества на любую из осей координат также является хорошим, после этого мы сможем стандартными трюками (замена $\forall$ на $\exists$ и элиминация кванторов, начиная с самого вложенного) доказать текущую теорему.

Также аналогичным образом вводим диаграммы, но в этот раз в диаграмму включаем только корни многочленов (потому что никаких «промежуточных значений» нет), состояния клеток — «равно нулю» и «не равно нулю». Корни теперь можно записывать в любом порядке. Более того, мы для любого многочлена знаем, сколько корней нужно добавить: ровно его степень минус количество уже существующих. ◀

2.2. Гильбертовское исчисление высказываний и предикатов

Смысл происходящего: сейчас хотим чисто синтаксическими преобразованиями научиться выводить все истинные формулы. Сначала — предикатные, но это при желании расширяется.

Def 2.2.1. Предикатная формула называется *тавтологией*, если она истинна в любой интерпретации.

Замечание 2.2.1. Пусть есть набор аксиом (возможно, бесконечный): $A_1, \dots, A_i, \dots$ и формула φ. Тогда естественно считать, что тавтологичность $(\bigwedge A_i) \to \varphi$ эквивалентна тому, что φ выводится из аксиом (здесь и далее $\to$ — следствие, являющееся частью предикатной формулы).

Теорема 2.2.1. Если в сигнатуре достаточно много предикатных и функциональных символов, то множество тавтологий этой сигнатуры неразрешимо.

▶ **TODO** на практике (что-то про ассоциативность) ◀

Теорема 2.2.2 (Гёделя о полноте исчисления предикатов)**.** Для любой перечислимой сигнатуры множество тавтологий перечислимо.

Замечание 2.2.2. К этой теореме мы будем долго идти и в конце концов докажем более сильное утверждение.

Def 2.2.2. Правило вывода в Гильбертовском исчислении высказываний (modus ponens):

$$\frac{A \to B, A}{B}$$

Читать так: если есть формулы $A \to B$ и A, то можно вывести формулу B.

Список аксиом (их много, но на экзамене можно будет сделать шпаргалку с ними):

1. $A \to (B \to A)$ — истину всегда можно вывести
2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ — некоторая транзитивность: если при условии A из B можно вывести C, то если выведем B из A, то и C из A тоже выведем.
3. $(A \wedge B) \to A$
4. $(A \wedge B) \to B$
5. $A \to (B \to (A \wedge B))$ — при условии, что верно и A, и B, верно $A \wedge B$.
6. $A \to (A \vee B)$
7. $B \to (A \vee B)$
8. $(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$ — если C выводится и из A, и из B, то выводится из $A \vee B$.
9. $\neg A \to (A \to B)$ — из лжи следует всё, что угодно
10. $(A \to B) \to ((A \to \neg B) \to \neg A)$ — из A не могут одновременно следовать и B, и $\neg B$
11. $A \vee \neg A$

Все эти формулы являются тавтологиями.

Def 2.2.3. Формула φ *выводима*, если существует последовательность формул $C_1, \dots, C_k$ такая, что:

- $C_k = \varphi$
- Формула C_i получена одним из двух способов:
  1. Аксиома, в которую подставлены произвольные формулы вместо переменных
  2. Результат modus ponens для каких-то двух формул C_a и C_b, где $a, b < i$.

Теорема 2.2.3 (корректность)**.** Если φ выводима, то φ — тавтология.

▶ Все аксиомы тавтологичны: надо просто рассмотреть таблицу истинности для каждой из них. Так как от интерпретации у нас зависят только изначальные значения переменных, то этого хватит.

Использование modus ponens позволяет нам из двух тавтологий получить некоторую третью формулу, которая тоже является тавтологий (опять же, рассмотрим таблицу истинности), что и требовалось. ◀

Лемма 2.2.1. $\mathcal{A} \to \mathcal{A}$ выводима

▶ Рассмотрим вторую аксиому, подставим в неё $A = \mathcal{A}$, $C = \mathcal{A}$, $B = \mathcal{A} \to \mathcal{A}$:

$$(\underbrace{\mathcal{A}}_{A} \to (\underbrace{(\mathcal{A} \to \mathcal{A})}_{B} \to \underbrace{\mathcal{A}}_{C})) \to ((\underbrace{\mathcal{A}}_{A} \to \underbrace{(\mathcal{A} \to \mathcal{A})}_{B}) \to (\underbrace{\mathcal{A}}_{A} \to \underbrace{\mathcal{A}}_{C}))$$

Теперь отдельно рассмотрим первую аксиому, подставим $A = \mathcal{A}$ и $B = \mathcal{A} \to \mathcal{A}$:

$$\underbrace{\mathcal{A}}_{A} \to (\underbrace{(\mathcal{A} \to \mathcal{A})}_{B} \to \underbrace{\mathcal{A}}_{A})$$

Теперь по modus ponens соединим эти две формулы, получим:

$$(\mathcal{A} \to (\mathcal{A} \to \mathcal{A})) \to (\mathcal{A} \to \mathcal{A})$$

Теперь заметим, что первая половина этой формулы — аксиома 1 с подставленными $A = B = \mathcal{A}$, то есть тавтология. Применяем modus ponens еще раз:

$$\mathcal{A} \to \mathcal{A}$$

◀

2.2.1. Натуральный вывод

Пусть Γ — список пропозициональных формул.

Def 2.2.4. Говорим, что $\Gamma \vdash \varphi$ (φ *выводима из* Γ), если есть конечная последовательность формул $C_1, \dots, C_k$ такая, что $C_k = \varphi$ и каждое C_i получено одним из способов:

- $C_i \in \Gamma$
- C_i получается из аксиом произвольной подстановкой
- C_i получается по modus ponens из каких-то предыдущих C_j, C_k $(j, k < i)$.

Лемма 2.2.2 (о дедукции)**.** $\Gamma \vdash (A \to B) \iff \Gamma, A \vdash B$

▶ **⇒:** Пусть у нас есть Γ, A. Выведем $(A \to B)$, используя только Γ (так можно сделать — это левая часть леммы). Теперь у нас есть A и $A \to B$, применим modus ponens, получим B.

⇐: Рассмотрим вывод B из Γ, A — это какие-то C_1, C_2, ..., $C_k = B$. Рассмотрим следующую последовательность:

$$(A \to C_1); (A \to C_2); \dots; (A \to C_k) = (A \to B)$$

Это очень похоже на вывод формулы $A \to B$, в частности, последний элемент является тем, чем надо. Давайте разберёмся с остальными индукцией по k, нам надо для каждого элемента $A \to C_i$ показать, что надо добавить перед ним, чтобы получить корректный вывод в Γ.

- Если $C_i = A$, то формула $A \to A$ просто является тавтологией, её мы выводить умеем по лемме 2.2.1
- Если $C_i \in \Gamma$, то мы можем вывести $A \to C_i$:

$$\begin{array}{cc} C_i \in \Gamma & \\ C_i \to (A \to (C_i)) & \text{аксиома 1} \\ A \to C_i & \text{modus ponens} \end{array}$$

- Если C_i — аксиома, то можно поступить аналогично предыдущему пункту
- Если C_i — modus ponens от C_x и C_y $(x, y < i)$. Переобозначим: $C_x = E \to C_i$, $C_y = E$ (вид именно такой, так как мы применили modus ponens). И теперь выведем:

$$\begin{array}{cc} A \to (E \to C_i) & \text{выведено раньше} \\ A \to E & \text{выведено раньше} \\ (A \to (E \to C_i)) \to ((A \to E) \to (A \to C_i)) & \text{аксиома 2 с подстановкой} \\ (A \to E) \to (A \to C_i) & \text{modus ponens} \\ A \to C_i & \text{modus ponens} \end{array}$$

◀

Заметим, что мы в процессе доказательства леммы пользовались только аксиомами 1 и 2. Вообще говоря, лемма о дедукции намного удобнее этих аксиом.

Пример 2.2.1. Аксиомы 1 и 2 можно вывести из леммы о дедукции.

▶ 1. Пусть есть конкретные формулы A и B, докажем аксиому 1 через лемму о дедукции:

$$\Gamma \vdash (A \to (B \to A)) \iff \Gamma, A \vdash B \to A \iff \Gamma, A, B \vdash A$$

2. Пусть есть конкретные формулы A, B и C, докажем аксиому 2 через лемму о дедукции:

$$\begin{array}{c} \Gamma \vdash (A \to (B \to C)) \to ((A \to B) \to (A \to C)) \\ \Updownarrow \\ \Gamma, (A \to (B \to C)) \vdash ((A \to B) \to (A \to C)) \\ \Updownarrow \\ \Gamma, (A \to (B \to C)), (A \to B) \vdash (A \to C) \\ \Updownarrow \\ \Gamma, (A \to (B \to C)), (A \to B), A \vdash C \end{array}$$

Применяем modus ponens в последней строчке (с посылкой A), получаем сначала $B \to C$, потом B, потом modus ponens к ним и получаем C.

◀

Теперь мы хотим аналогичным образом написать какие-то более удобные правила для доказательств, аналогичные оставшимся аксиомам.

№	Аксиома	Правило вывода
3	$(A \wedge B) \to A$	$\Gamma, (A \wedge B) \vdash A$
4	$(A \wedge B) \to B$	$\Gamma, (A \wedge B) \vdash B$
5	$(A \to (B \to (A \wedge B)) \to B$	Если $\Gamma \vdash A$ и $\Gamma \vdash B$, то $\Gamma \vdash (A \wedge B)$
6	$(A \to (B \vee A))$	$\Gamma, A \vdash (B \vee A)$
7	$(A \to (B \vee A))$	$\Gamma, B \vdash (B \vee A)$
8	$(A \to C) \to ((B \to C) \to ((A \vee B) \to C))$	Если $\Gamma, A \vdash C$ и $\Gamma, B \vdash C$, то $\Gamma, (A \vee B) \vdash C$
9	$\neg A \to (A \to B)$	Если $\Gamma \vdash A$ и $\Gamma \vdash \neg A$, то $\Gamma \vdash B$
10	$(A \to B) \to ((A \to \neg B) \to \neg A)$	Если $\Gamma, A \vdash B$ и $\Gamma, A \vdash \neg B$, то $\Gamma \vdash \neg A$
11	$(A \vee \neg A)$	Если $\Gamma, A \vdash B$ и $\Gamma, \neg A \vdash B$, то $\Gamma \vdash B$

Теперь правило у аксиомы 10 показывает, что рассуждение «от противного» работает и в Гильбертовском исчислении: если хотим показать $\Gamma \vdash \neg X$, то надо предположить X и вывести из Γ, X два утверждения: B и $\neg B$, тогда по правилу 10 получим X,

А правило для аксиомы 11 требует некоторых пояснений:

▶ Пусть $\Gamma, A \vdash B$ и $\Gamma, \neg A \vdash B$. Тогда по правилу для восьмой аксиомы получаем, что $\Gamma, (A \vee \neg A) \vdash B$. А вот $A \vee \neg A$ — это как раз аксиома, то есть её можно из левой части убрать, получить $\Gamma \vdash B$. ◀

Лемма 2.2.3 (снятие двойного отрицания)**.** $\vdash \neg\neg\mathcal{A} \to \mathcal{A}$ (следует из «ничего», то есть из аксиом)

▶ Будем преобразовывать:

$$\vdash \neg\neg\mathcal{A} \to \mathcal{A} \iff \neg\neg\mathcal{A} \vdash \mathcal{A}$$
$$\Updownarrow \text{ добавили аксиому 11 с подстановкой}$$
$$\neg\neg\mathcal{A}, \mathcal{A} \vee \neg\mathcal{A} \vdash \mathcal{A}$$
$$\Updownarrow$$
$$\neg\neg\mathcal{A} \vdash (\mathcal{A} \vee \neg\mathcal{A}) \to \mathcal{A}$$
$$\Uparrow \text{ по аксиоме 8}$$
$$\neg\neg\mathcal{A} \vdash (\mathcal{A} \to \mathcal{A}), (\neg\mathcal{A} \to \mathcal{A}) \iff$$
$$\iff \begin{cases} \neg\neg\mathcal{A}, \mathcal{A} \vdash \mathcal{A} & \text{очевидно} \\ \neg\neg\mathcal{A}, \neg\mathcal{A} \vdash \mathcal{A} & \text{верно, так как слева есть и } \neg\mathcal{A}\text{, и его отрицание, т.е. выводится что угодно} \end{cases}$$

◀

Замечание 2.2.3. Теперь рассуждение «от противного» действительно рассуждение «от противного»: предполагаем $\neg X$, выводим противоречие, делаем вывод X (нам для этого нужно снятие двоного отрицания).

Пример 2.2.2. Докажем $(A \to B) \to (\neg B \to \neg A)$.

▶ Перепишем по лемме о дедукции:

$$(A \to B) \overset{?}{\vdash} (\neg B \to \neg A)$$
$$\Updownarrow$$
$$A \to B, \neg B \overset{?}{\vdash} \neg A$$

Рассуждаем от противного (по переформулировке аксиомы 10). Добавляем A к предпосылкам и выводим B и $\neg B$:

$$A \to B, \neg B, \neg\neg A$$
применяем снятие двойного отрицания
$$A \to B, \neg B, \neg\neg A, A$$
применяем modus ponens
$$A \to B, \neg B, \neg\neg A, B$$

Успех: вывели и $\neg B$, и B. ◀

Def 2.2.5. Список формул Γ противоречив, если $\Gamma \vdash B, \neg B$ для некоторого B.

Def 2.2.6. Список формул Γ совместен, если есть значения переменных, которые выполнит все формулы из Γ.

Теорема 2.2.4 (теорема о корректности и полноте в сильной форме)**.** Γ совместно $\iff$ Γ непротиворечиво.

Замечание 2.2.4. Стрелка $\Rightarrow$ очевидна: если это не так, что при некоторых значениях формул верны все формулы из Γ и все, которые можно вывести, то есть B и $\neg B$, но они одновременно верны быть не могут.

Следствие 2.2.4.1. Если φ — тавтология, то из пустого списка можно вывести φ.

▶ Доказательство от противного (пример 2.2.2). Предположим $\neg\varphi$ и выведем два утверждения: B и $\neg B$ (т.е. покажем $\neg\varphi \vdash B$ и $\neg\varphi \vdash \neg B$.

Так как φ — тавтология, то нет значений переменных, при которых $\neg\varphi$ верна. То есть список $\Gamma = \{\neg\varphi\}$ не является совместным. Тогда по теореме Γ противоречив, то есть можно вывести и B, и $\neg B$.

Таким образом получаем $\vdash \neg\neg\varphi$, а снимать двойное отрицание мы уже умеем и получим $\vdash \varphi$. ◀

2.2.2. Доказательство теоремы о полноте

Def 2.2.7. Γ является полным, если для любой формулы φ, использующей те же переменные, что и Γ либо $\Gamma \vdash \varphi$, либо $\Gamma \vdash \neg\varphi$.

Лемма 2.2.4. Любой непротиворечивый список Γ можно расширить до полного непротиворечивого.

▶

Утверждение 2.2.1. Пусть Γ непротиворечив и $\Gamma \nvdash \neg\varphi$, тогда Γ, φ непротиворечив.

▶ От противного: пусть противоречиво, т.е. можно вывести и A, и $\neg A$. Тогда отсюда знаем, что $\Gamma \vdash \neg\varphi$, противоречие. ◀

Теперь доказываем лемму в конечном/счётном случае. Переменных у нас тогда тоже конечно/счётно, соответственно, формул у нас тоже конечно/счётно. Добавляем по одной: берём формулу φ, если нельзя вывести ни её, ни отрицание, то добавляем её в Γ. За каждый шаг получаем непротиворечивое.

Пусть Γ' — это объединение всех Γ по шагам (этакий предел). Если оно противоречиво, то противоречие выводится за конечное число шагов, использует конечное число формул, т.е. противоречие выводится уже на каком-то конечном шаге алгоритма, чего быть не может.

Доказательство для несчётного Γ пока оставляется на конец курса — нам потребуется аксиома выбора/лемма Цорна. ◀

Теперь доказываем теорему 2.2.4 (стрелку $\Leftarrow$). Был непротиворечивый Γ, расширили до полного непротиворечивого Γ', теперь для любой переменной x выводится либо x, либо $\neg x$. Теперь подставим значения: $x = 1$, если выводится x, и $x = 0$, если выводится $\neg x$.

Утверждение 2.2.2. Все выводимые формулы в такой подстановке истинны, а все невыводимые — ложны.

▶ Индукция по построению формулы. База очевидна: это просто переменные. Для перехода надо разобраться со связками:

- Отрицание. Пусть знаем для φ что её выводимость из Γ' эквивалентна истинности в подстановке ($\Gamma' \vdash \varphi \iff \varphi = 1$). Теперь рассмотрим формулу $\neg\varphi$. Так как Γ' непротиворечиво (обе быть выводимы не могут) и полно (хотя бы одно выводится): $\Gamma' \vdash \neg\varphi \iff \Gamma' \nvdash \varphi \iff \varphi = 0 \iff \neg\varphi = 1$.
- Конъюнкция. Для доказательства перехода надо доказать: $\Gamma' \vdash \varphi \wedge \psi \iff (\Gamma' \vdash \varphi) \wedge (\Gamma' \vdash \psi)$.

  $\Rightarrow$**:** Очевидно, так как есть две аксиомы.

$\Leftarrow$: Было правило, переформулированное из аксиомы 3.

- Дизъюнкция. Надо доказать, что:

$$\Gamma' \vdash \varphi \vee \psi \iff \left[\begin{array}{l} \Gamma' \vdash \varphi \\ \Gamma' \vdash \psi \end{array}\right.$$

$\Leftarrow$: Были аксиомы.

$\Rightarrow$: Доказательство от противного, интересный случай: $\varphi \vee \psi$ выводится, а вот φ и ψ — нет (т.е. по полноте Γ выводятся $\neg\varphi$ и $\neg\psi$). Давайте покажем противоречивость $\Gamma', \varphi \vee \psi$: выведем из неё φ и $\neg\varphi$. Из них $\neg\varphi$ выводится по предположению. А дальше есть правило для дизъюнкции, т.е. надо проверить, что φ выводится в двух случаях:

– Γ', φ — берёт и выводится сразу

– Γ', ψ — из Γ' выводится $\neg\psi$, т.е. Γ', ψ противоречиво и можно вывести что угодно, в том числе φ.

Таким образом, $\Gamma', \varphi \vee \psi$ противоречиво, что и требовалось. Значит, $\varphi \vee \psi$ выводится.

- Импликация.

Упражнение 2.2.1.

$$\Gamma' \vdash (\varphi \to \psi) \iff \left[\begin{array}{l} \Gamma' \nvdash \varphi \\ \Gamma' \vdash \psi \end{array}\right.$$

◀

2.2.3. Исчисление предикатов

Кажется, что-то похожее уже было на прошлой лекции, но всё равно повторим определение:

Def 2.2.8. Предикатная формула называется *общезначной* (или, реже, *тавтологией*), если она истина во всех интерпретациях и во всех оценках свободных переменных.

Список аксиом в исчислении предикатов: 11 старых плюс две новые аксиомы (для кванторов, иначе бы у нас кванторы навечно прилепились к атомарным формулам, что неинтересно, вывелось бы не всё, что хочется):

12. $(\forall x\colon \varphi(x)) \to \varphi(t|x)$
13. $\varphi(t|x) \to (\exists x\colon \varphi(x))$

Здесь $\varphi(t|x)$ — замена всех вхождений переменной x (разумеется, если какой-то квантор x не перекрыл) на терм t, свободный для подстановки (то есть свободные переменные терма t не попадают в область действия кванторов φ, куда бы мы не подставили t).

Пример 2.2.3.

$$\forall y\colon (P(x,y) \to Q(x,y))$$

Подстановка $y|x$ (вместо x подставляем y) — не свободная, потому что y — свободная переменная, а в выражении есть квантор по y.

Пример 2.2.4.

$$\begin{gathered} R(x) \wedge (\forall x\colon (P(x,y) \to Q(x,y))) \\ \downarrow (y \vee z)|x \\ R(y \vee z) \wedge (\forall x\colon (P(x,y) \to Q(x,y))) \end{gathered}$$

Внутрь квантора по x мы не залезаем.

Пример 2.2.5. Всегда можно подставлять $t = x$, а также $t = c$, где c — новая константа в смысле «функциональный символ арности ноль».

Два новых правила вывода (modus ponens остаётся), правила Бернайса:

1. $\frac{\varphi \to \psi}{\varphi \to \forall x \colon \psi}$, если x не входит в свободные переменные φ.

   Смысл такой: если у ψ есть параметр x, а $\varphi \to \varphi$ верно независимо от этого параметра, то если φ, то ψ тоже верно независимо от этого параметра.

   Пример 2.2.6. Из формулы $(y \vee z) \to (x \wedge y)$ можно вывести $(y \vee z) \to (\forall x \colon (x \wedge y))$.

2. $\frac{\varphi \to \psi}{(\exists x \colon \varphi) \to \psi}$, если x не входит в свободные переменные ψ.

   Смысл такой: если у φ есть параметр x, а у ψ его нет и при этом из φ всегда следует ψ, то если φ верна хоть при каком-то параметре, то ψ тоже верна.

   Пример 2.2.7. Из формулы $(x \wedge y) \to (y \vee z)$ можно вывести $(\exists x \colon (x \wedge y)) \to (y \vee z)$. То есть если хоть при каком-то x будет верно $x \wedge y$, то $y \vee z$ тоже будет верно.

Def 2.2.9. Формула φ *выводима*, если существует последовательность предикатных формул $A_1, \dots, A_k$ такая, что:

- $A_k = \varphi$
- Формула A_i получена одним из двух способов:
  1. Аксиома 1–13, в которую подставлены произвольные формулы вместо переменных
  2. Результат правил вывода для каких-то двух формул A_a и A_b, где $a, b < i$.

Теорема 2.2.5 (о корректности)**.** φ выводима $\Rightarrow$ φ общезначна.

▶ Для аксиом 1–11 общезначность мы знали, надо проверить для двух новых. Это упражнение. И еще надо проверить для правил Бернайса:

1. Так как x не входит в левую часть и формула сверху общезначна, то независимо от значения x верно $\varphi \to \psi$, при этом φ от x вообще не зависит, т.е. $\varphi \to \forall x \colon \psi$.
2. Аналогично: надо посмотреть на случай, когда $\varphi \to \psi$ и $\exists x \colon \varphi$. Берём этот конкретный x, подставляем в φ, отсюда немедленно получаем корректность ψ.

◀

Пример 2.2.8. Примеры выводимых формул:

1. Все, полученные из пропозициональных тавтологий подстановкой предикатных формул вместо переменных

2. Если выводима $\varphi \to \psi$, то выводима *контрпозиция*: $\neg\psi \to \neg\varphi$.

   ▶ Такая штука просто является пропозициональной тавтологией: $(\varphi \to \psi) \to (\neg\psi \to \neg\varphi)$. Тогда применяем modus ponens и получаем, что надо. ◀

3. $\varphi \to \psi$, $\psi \to \tau$ выводимы, тогда $\psi \to \tau$ выводима.

   ▶ Была пропозициональная тавтология $(\varphi \to \psi) \to ((\psi \to \tau) \to (\varphi \to \tau))$, применяем два раза modus ponens ◀

4. $(\forall x \colon \varphi(x)) \to (\exists x \colon \varphi(x))$

   ▶ Есть аксиома $(\forall x \colon \varphi(x)) \to \varphi(x)$, есть аксиома $\varphi(x) \to (\exists x \colon \varphi(x))$, дальше по пункту 3. ◀

5. $(\exists y \colon \forall x \colon \varphi) \to (\forall x \colon \exists y \colon \varphi)$

▶ Есть аксиома $(\forall x\colon \varphi) \to \varphi$, есть аксиома $\varphi \to (\exists y\colon \varphi)$, по пункту 3 собираем $(\forall x\colon \varphi) \to (\exists y\colon \varphi)$.
Теперь применяем правила Бернайса:

$$\underbrace{(\forall x\colon \varphi)} \to \underbrace{(\exists y\colon \varphi)}$$

второе правило Бернайса для переменной y

$$(\exists y\colon \overbrace{\forall x\colon \varphi}) \to \overbrace{(\exists y\colon \varphi)}$$

$$\underbrace{(\exists y\colon \forall x\colon \varphi)} \to \underbrace{(\exists y\colon \varphi)}$$

первое правило Бернайса для переменной x

$$\overbrace{(\exists y\colon \forall x\colon \varphi)} \to \forall x\colon \overbrace{(\exists y\colon \varphi)}$$

Что и требовалось. ◀

6. $\neg(\forall x\colon \varphi(x)) \to (\exists \neg\varphi(x))$.

▶ Воспользуемся контрпозицией. Достаточно вывести:

$$\neg(\exists x\colon \neg\varphi(x)) \to (\forall x\colon \varphi(x))$$

Для этого достаточно вывести без квантора $\forall x$ справа — его всегда можно добавить при помощи первого правила Бернайса (так как x не входит в свободные переменные левой части).

$$\neg(\exists x\colon \neg\varphi(x)) \to \varphi(x)$$
$$\Updownarrow \text{ применяем контрпозицию}$$
$$\neg\varphi(x) \to (\exists x\colon \neg\varphi(x))$$

А это уже аксиома, получили требуемое доказательство. ◀

Def 2.2.10. Пусть Γ — список замкнутых формул. Называем формулу A *выводимой* (и обозначаем $\Gamma \vdash A$), если её можно вывести, используя наравне с аксиомами формулы из Γ (важно, что они замкнытые).

Упражнение 2.2.2. Показать, что если $A(x)$ выводимо, то и $\forall x\colon A(x)$ выводимо.

Замечание 2.2.5. Формула $A(x) \to \forall x\colon A(x)$ не является общезначимой: действительно, если положить $A(x) \iff x = 0$, то в оценке свободной переменной $x = 0$ левая часть верна, а вот правая — ложна.

Лемма 2.2.5 (о дедукции)**.** Если Γ — список замкнутых формул, φ замкнута, тогда:

$$\Gamma \vdash \varphi \to \psi \iff \Gamma, \varphi \vdash \psi$$

▶ **⇒:** очевидно, так как можно вывести $\varphi \to \psi$, а φ и modus ponens у нас уже есть.

⇐: Берём какой-то вывод: $A_1, \dots, A_k = \psi$. Убираем φ из предпосылок, дописываем φ ко всем формулам $(\varphi \to A_i)$, надо показать, что это можно достроить до вывода. В случае, когда мы для получения очередной формулы использовали modus ponens всё идёт аналогично случаю для пропозициональных формул.
Покажем, что делать в случае, когда использовали первое правило Бернайса для вывода. Хотим показать, что вот такое верно (тут x не входит в свободные переменные A):

$$\frac{\varphi \to (A \to B)}{\varphi \to (A \to \forall x\colon B)}$$

То, что сверху, пропозиционально эквивалентно $(\varphi \wedge A) \to B$, отсюда по правилу Бернайса $(\varphi \wedge A) \to (\forall x\colon B)$ (так как φ вообще замкнута), что пропозиционально эквивалентно $\varphi \to (A \to \forall x\colon B)$.

Замечание 2.2.6. «Пропозиционально эквивалентно» — взяли пропозициональную аксиому, подставили в неё нужные переменные, применили modus ponens.

Второе правило Бернайса: хотим показать, что вот такое можно достроить до вывода (тут x не входит в свободные переменные B):

$$\frac{\varphi \to (A \to B)}{\varphi \to (\exists x\colon A) \to B}$$

То, что сверху, пропозиционально эквивалентно $A \to (\varphi \to B))$, применяем второе правило Бернайса:

$$(\exists x\colon A) \to (\varphi \to B)$$
пропозиционально эквивалентно
$$\varphi \to ((\exists x\colon A) \to B)$$

◀

Свойства выводимости из списка:

1. Если $\Gamma \vdash A$, $\Gamma' \supseteq \Gamma$, то $\Gamma' \vdash A$.
2. Если $\Gamma \vdash A$, то существует конечное $\Gamma' \subseteq \Gamma$ такое, что $\Gamma' \vdash A$ (так как в выводе участвует лишь конечное число формул).
3. $\gamma_1, \gamma_2, \dots, \gamma_n \vdash A$ (все γ_i замкнуты), тогда выводимы формулы $\gamma_1 \to (\gamma_2 \to (\dots (\gamma_n \to A) \dots))$ и $(\gamma_1 \wedge \gamma_2 \wedge \dots \wedge \gamma_n) \to A$.

2.2.4. Полнота исчисления предикатов в сильной форме

Лемма 2.2.6 (о свежих константах)**.** Пусть $\Gamma \vdash \varphi(c|x)$, где c — некоторая константа (нульместный функциональный символ), не входящая ни в φ, ни в Γ. Тогда $\Gamma \vdash \varphi$.

▶ Рассмотрим вывод $\varphi(c|x)$. Заменим в выводе c на y, где y — какая-то новая свободная переменная, получили корректный вывод формулы $\varphi(y|x)$. Потом по упражнению 2.2.2 выведем отсюда:

$$\forall y\colon \varphi(y|x)$$

Теперь берём аксиому 12:

$$(\forall y\colon \varphi(y|x)) \to \varphi(x|(y|x))$$

Теперь применяем modus ponens, получаем просто φ, что и требовалось. ◀

Лемма 2.2.7 (о добавлении константы)**.** Пусть φ — формула сигнатуры σ, и она выводима в сигнатуре σ', которая отличается от σ добавлением константы c, не содержащейся ни в φ, ни в σ. Тогда φ выводима и в σ.

▶ Возьмём эту новую константу c, она где-то в выводе встречается. Заменим её на какую-нибудь новую переменную y. Заметим, что вывод всё еще остался корректным, причём в конце мы всё еще получим φ (так как эта константа в φ не содержалась). ◀

Def 2.2.11. Пусть Γ — список замкнутых формул. Тогда интерпретация I называется *моделью* для Γ, если она выполняет все формулы из Γ.

Напоминаем, что интерпретация — это множество-носитель интепретации плюс задание всех операций из сигнатуры на этом носителе.

Теорема 2.2.6 (о полноте в сильной форме)**.** Список формул Γ непротиворечив $\iff$ Γ имеет модель.

▶ **$\Leftarrow$:** по ходу вывода модель сохраняется (то есть все новые формулы всё еще выполняются): надо это проверить для всех аксиом, потом для правил вывода.

$\Rightarrow$:

Def 2.2.12. Γ — *полный*, если для любой замкнутой формулы φ либо $\Gamma \vdash \varphi$, либо $\Gamma \vdash \neg\varphi$.

Лемма 2.2.8. Непротиворечивый Γ можно непротиворечиво расширить до полного списка. Доказательство аналогично доказательству леммы 2.2.4 (у нас тогда еще не было предикатов, но они ничего не портят).

Def 2.2.13. Список Γ — *экзистенциально полный*, если для любой замкнутой формулы вида $\exists x\colon \varphi$ верно, что если $\Gamma \vdash (\exists x\colon \varphi)$, то найдётся замкнутый терм t такой, что $\Gamma \vdash \varphi(t|x)$.

Лемма 2.2.9. Непротиворечивый Γ можно непротиворечиво расширить до полного и экзистенциально полного списка.

▶ Сначала продолжим Γ до полного Γ'. Теперь выводится любая формула или её отрицание, но экзистенциальной полноты может не быть.

Давайте если $\Gamma' \vdash (\exists x\colon \varphi)$, то добавим формулу $\varphi(c)$, где c — новая константа. Противоречивость не образуется: если образуется, то из некоторой формулы γ (конечная конъюнкция формул из Γ', участвующих в выводу) было выводимо $\neg\varphi(c)$, а по лемме 2.2.6 (о свежих константах) получаем $\gamma \to \neg\varphi$. Делаем контрапозицию, получаем $\varphi \to \neg\gamma$. Применяем правило Бернайса для переменной x (формула γ замкнутая), получаем $(\exists x\colon \varphi) \to \neg\gamma$. Получаем, что множество Γ' было противоречиво изначально.

Дальше будем на одном шаге делать экзистенциальное пополнение, потом обычное, и так чередуем, получаем какую-то последовательность вложенных Γ'_i. Возьмём их объединение, получим искомый список. В самом деле: он непротиворечив (потому что противоречие выводится из конечного шагов и из конечного числа Γ'_i), он полон (так как каждая формула φ содержит конечное число констант, которые мы добавляем в Γ' и тогда на каком-то шаге она или её отрицание станут выводимы), экзистенциально полно по той же причине. ◀

Пусть σ — сигнатура Γ' (экзистенциально расширенного списка Γ). Скажем, что носитель модели — это множество замкнутых термов сигнатуры σ. Теперь надо определить функции и предикаты.

Замечание 2.2.7. Осторожно, не запутайтесь дальше (есть шанс, что автор конспекта сам запутался): когда мы говорим «терм» и подставляем термы, то мы можем это делать в двух смыслах: либо «терм» в смысле «элемент носителя» и подстановки переменной, либо нормальная подстановка некоторого терма в формулу.

Рассмотрим функцию $f^{(n)} \in \sigma$ из сигнатуры. Капитан Очевидность предлагает определить их вот так:

$$[f^{(n)}](t_1, \dots, t_n) = f^{(n)}(t_1, \dots, t_n)$$

То есть если применяем функциональный символ к термам $t_1, \dots, t_n$, то мы получаем терм вида «$f^{(n)}(t_1, \dots, t_n)$» (он тоже вполне себе замкнутый).

Теперь возьмём предикатный символ $p^{(k)} \in \sigma$. Определим его значения на термах так:

$$[p^{(k)}](t_1, \dots, t_n) = \begin{cases} 1, & \text{если } \Gamma' \vdash p^{(k)}(t_1, \dots, t_n) \\ 0, & \text{если } \Gamma' \vdash \neg p^{(k)}(t_1, \dots, t_n) \end{cases}$$

Так как Γ' полон, то все значения определены.
Получили некоторую интерпретацию I: носитель и формулы.

Утверждение 2.2.3. Для любой замкнутой формулы φ сигнатуры σ верно: $\Gamma' \vdash \varphi \iff \varphi$ истинна в интерпретации I.

▶ Индукция по числу логических операций в φ ($\wedge, \vee, \neg, \to, \exists, \forall$). База: атомарные формулы. Для них мы просто по определению I всё корректно задали.
Переход: надо разобраться со значками (доказательство аналогично утверждению 2.2.2, там мы пользовались только полнотой, даже экзистенциальность не нужна):
- $\Gamma' \vdash (\varphi \wedge \psi) \iff \Gamma' \vdash \varphi$ и $\Gamma' \vdash \psi$
- $\Gamma' \vdash (\varphi \vee \psi) \iff \Gamma' \vdash \varphi$ или $\Gamma' \vdash \psi$
- $\Gamma' \vdash \varphi \iff \Gamma' \nvdash \varphi$
- $\Gamma' \vdash (\varphi \to \psi) \iff \Gamma' \nvdash \varphi$ или $\Gamma' \vdash \psi$

Остались кванторы.

$\exists$ Допустим, что $\Gamma' \vdash \exists\colon \varphi(x)$. Надо доказать, что эта формула истинна в нашей интерпретации. Мы знаем, что имеется экзистенциальная полнота Γ', то есть имеется такой замкнутый терм t такой, что $\Gamma \vdash \varphi(t|x)$. По предположению индукции $\varphi(t|x)$ истинна (так как квантор сняли, а никаких новых связок не добавили), что и требовалось.
Теперь в обратную сторону: пусть $\exists x\colon \varphi(x)$ истинна в интерпретации I, тогда существует некоторое значение x (назовём его термом t) такой, что $\varphi(t|x)$ истинна. То есть $\Gamma' \vdash \varphi(t|x)$. Есть аксиома: $\varphi(t|x) \to \exists x\colon \varphi(x)$, По modus ponens аксиомы с $\varphi(t|x)$ получаем $\Gamma' \vdash \exists x\colon \varphi(x)$, что и требовалось.

$\forall$ Пусть $\Gamma' \vdash (\forall x\colon \varphi(x))$. Есть аксиома: $(\forall x\colon \varphi(x)) \to \varphi(t|x)$. То есть для любого терма t верно: $\Gamma' \vdash \varphi(t|x)$ (по modus ponens), то есть $\varphi(t|x)$ истинна, откуда $\forall x\colon \varphi(x)$ истинна.
В обратную сторону: пусть $\forall x\colon \varphi(x)$ истинна в I. Доказательство от противного: пусть $\Gamma' \nvdash (\forall x\colon \varphi(x))$, тогда $\Gamma' \vdash \neg(\forall x\colon \varphi(x))$. Мы уже проверяли, что выводима формула $\neg(\forall x\colon \varphi(x)) \to (\exists x\colon \neg\varphi(x))$. По modus ponens выводим $\exists x\colon \neg\varphi(x)$. Теперь по предыдущему пункту доказательства можем снять $\exists$ и обнаружить, что существует такой замкнутый терм t, что $\neg\varphi(t|x)$ истинна, то есть $\varphi(t|x)$ ложна. Отсюда $\forall x\colon \varphi(x)$ ложно (так как существует терм x, для которого подстановка ложна), противоречие.

◀
◀

Следствие 2.2.6.1. Если φ общезначна, то φ выводима.

▶ Введём $\tilde{\varphi}$ — заменяем все свободные переменные на новые константы (это чисто синтаксическая операция: никаких носителей еще нет, никаких значений функций на них задавать не надо). Выводимость от этого не меняется. Теперь рассмотрим список из одной (теперь уже замкнутой) формулы $\{\neg\tilde{\varphi}\}$. Он либо противоречив, либо непротиворечив.
Если противоречив, то для какой-то A:

$$\begin{cases} \neg\tilde{\varphi} \vdash A \\ \neg\tilde{\varphi} \vdash \neg A \end{cases}$$

У нас было правило вывода для рассмотрения от противного: если из x выводится и A, и $\neg A$, то из пустого выводится $\neg x$. Еще было правило снятия двойного отрицания:

$$\begin{cases} \vdash \neg\neg\tilde{\varphi} \\ \neg\neg\tilde{\varphi} \vdash \tilde{\varphi} \end{cases} \Rightarrow (\vdash \tilde{\varphi})$$

Что и требовалось.

Если непротиворечив, то есть модель I. Однако в этой модели мы получаем, что $\tilde{\varphi}$ ложно, то есть она точно не общезначна. Значит, φ ложна, противоречие. ◀

Теорема 2.2.7 (о компактности исчисления предикатов)**.** *Γ — список замкнутых формул. Пусть любое конечное подмножество Γ имеет модель. Тогда Γ тоже имеет модель.*

▶ Пусть Γ не имеет модели. Тогда он противоречив, то есть можно вывести и некоторую формулу, и её отрицание. В выводе используется лишь конечное множество формул из Γ. То есть имеется некоторое конечное противоречивое подмножество Γ, которое, как мы знаем, не имеет модели. Противоречие. ◀

Замечание 2.2.8. Это некоторый удобный способ доказывать существования моделей у списков формул.

Замечание 2.2.9. Теперь мы знаем, что общезначимость формулы можно доказать с помощью формального вывода.

2.2.5. Подготовка к алгоритму проверки

Уже можно придумать полуразрешающий алгоритм для проверки формулы на тавтологичность, перебирающий все формальные выводы и проверяющий, не вывелась ли формула. Но это очень-очень долго.

Если у нас есть пропозициональная формула (без кванторов), то жить лучше: можно либо перебрать значения переменных, либо скачать SAT-Solver.

С кванторами хуже. Давайте сначала приведём формулу в предварённую (предварённую нормальную) форму, когда все кванторы идут в начале.

Формулы классифицируют по количеству *смен* кванторов, как в арифметической иерархии, например:

$$(\exists \dots \exists \colon \varphi) \in \Sigma_1$$
$$(\forall \dots \forall \colon \varphi) \in \Pi_1$$
$$(\exists \dots \exists \forall \dots \forall \colon \varphi) \in \Sigma_2$$
$$(\forall \dots \forall \exists \dots \exists \colon \varphi) \in \Pi_2$$

Утверждение 2.2.4.

$$\forall x_1 \colon \dots \forall x_n \colon \varphi(x_1, \dots, x_n)$$

Эта формула общезначна тогда и только тогда, когда общезначна φ.

▶ Очевидно, наверное. ◀

Теорема 2.2.8. *Бескванторная формула φ общезначна $\iff$ φ — частный случай пропозициональной тавтологии (то есть мы берём пропозициональную тавтологию и подставляем вместо переменных какие-то куски бескванторной формулы).*

▶ **$\Leftarrow$:** Уже было в древних примерах: пропозициональные тавтологии — частный случай тавтологий в исчислении предикатов. Если возьмём какую-нибудь тавтологию (например, $x \vee \neg x$) и заменим переменные на что-то, то это всё еще останется истинным утверждением.

$\Rightarrow$**:** Выпишем все различные атомарные формулы (предикатный символ от каких-то функциональных), которые участвуют в бесквантворной формуле: $P_1, P_2, \dots, P_n$ (все P_i попарно различны как строки). Обозначим значение каждой атомарной формулы за некоторую булевскую переменную, получили какую-то пропозициональную формулу φ'. Если она тавтология, то доказали, что надо.

Пусть не тавтологие, то есть имеется какая-то подстановка α переменных, при которых значение φ' — ложь. Давайте придумаем интерпретацию формулы φ, при которой она вычислится в ложь. Носителем возьмём замкнутые термы, функциональные символы определим естественным образом (как и раньше). А вот предикатные символы определим так, чтобы значения P_i совпали с подстановкой α: мы так можем сделать, так как все P_i различны. Те значения предикатных символов, которые так не определились, определим произвольно — они ни на что не повлияют. Таким образом получили интерпретацию, в которой исходная формула φ ложна, то есть она не общезначна, противоречие.

Пример 2.2.9. Возьмём какую-нибудь бесквантворную формулу:

$$P_1(f(x), g(y)) \to ((P_1(f(x), g(y)) \wedge P_2(y))$$

У нас есть две разных атомарных формулы, введём переменные: $a = P_1(f(x), g(y))$, $b = P_2(y)$. Получили пропозициональную формулу:

$$a \to (a \wedge b)$$

Эта формула тавтологией не является, например, при $a = 1$ и $b = 0$ она неверна.

Тогда можно построить интерпретацию исходной формулы, в которой она будет неверна.

◀

Теорема 2.2.9 (Эрбрана)**.** Следующие утверждения эквивалентны:

1. Замкнутая Σ_1-формула (у которой в начале идут только $\exists$) от n переменных общезначна
2. Существует некоторое множество замкнутых термов:

$$\begin{matrix} t_1^{(1)} & \dots & t_n^{(1)} \\ t_1^{(2)} & \dots & t_n^{(2)} \\ \vdots & \ddots & \vdots \\ t_1^{(k)} & \dots & t_n^{(k)} \end{matrix}$$

   такая, что общезначна следующая конечная дизъюнкция:

$$\bigvee_{i=1}^{k} \varphi(t_1^{(i)} \mid x_1, t_2^{(i)} \mid x_2, \dots, t_n^{(i)} \mid x_n)$$

▶ $\Uparrow$**:** пусть имеется конечная дизъюнкция и набор термов. Возьмём произвольную интерпретацию формулы, давайте по очереди попробуем для каждого i взять такие значения $x_1, \dots, x_n$: $x_1 = t_1^{(i)}, x_2 = t_2^{(i)}, \dots$. Хотя бы один набор выполнит исходную формулу, что и требовалось.

$\Downarrow$: Рассморим бесконечное множество A формул $\{\neg\varphi(t_1 \mid x_1, t_2 \mid x_2, \dots, t_n \mid x_n)\}$ для всех возможных $t_1, \dots, t_n$ (где t_i — замкнутые термы). Два случая:

1. Множество A противоречиво. Значит, противоречие можно вывести из конечного числа формул, то есть конъюнкция некоторого конечного числа отрицаний φ есть противоречие. По правилу де Моргана можно вынести отрицание наружу, получить общезначную дизъюнкцию.

2. Множество A непротиворечиво, то есть совместно. Тогда существует модель I с носителем M, которая выполняет все формулы A. Давайте рассмотрим множество $M' \subseteq M$ — множество значений всех замкнутых термов. Можно сузить модель I до множества M' (так как функциональные символы из M' не выводят), получить I'.
А в I' формула $\exists x_1 \colon \ldots \exists x_n \colon \varphi(\ldots)$ ложна, так как любой x_i имеет значение какого-то терма, то есть $\varphi(\ldots)$ всегда будет являться формулой из модели I', а эта модель выполняет все $\neg\varphi(\ldots)$, то есть $\varphi(\ldots)$ точно ложна.

◀

2.2.6. Сколемизация

Def 2.2.14. Пусть была формула:

$$\forall x_1 \colon \exists x_2 \colon \forall x_3 \colon \exists x_4 \colon \ldots$$

Давайте вычеркнем $\exists x_2$ и заменим x_2 везде в формуле на $f_1(x_1)$, так как x_2 зависит только от x_1. x_4 заменим на $f_2(x_1, x_3)$ и так далее. Результат называется сколемизацией.

Здесь функции f_1, f_2 — какие-то новые, которых раньше не было.

Если квантор существования идёт в самом начале, то добавляется функция с нулём аргументов.

Утверждение 2.2.5. φ выполнима $\iff$ её сколемизация $\tilde{\varphi}$ выполнима. Заметим, что тут не эквивалентность, а выполнимость.

▶ Детали были оставлены в качестве упражнения. Идея: в конкретной интерпретации задаём значения функций $f_1, f_2, \ldots$ по индукции, после обработки первых кванторов значение функции — в точности то, что выполнит остаток формулы (оно либо существует из-за квантора всеобщности, либо задаём произвольным). ◀

Замечание 2.2.10. φ общезначна $\iff$ $\neg\varphi$ не выполнима $\iff$ $\widetilde{\neg\varphi}$ не выполнима $\iff$ $\underbrace{\neg\widetilde{\neg\varphi}}_{\in\Sigma_1}$ общезначна.

Тут мы свели определение общезначности к теореме Эрбрана. **TODO** да ладно?

2.3. Теории и модели

Def 2.3.1. Теория — это множество замкнутых формул.

Def 2.3.2. Модель называется *нормальной*, если имеется двухместный предикатный символ $=$, обозначающий точное совпадение элементов носителя интерпретации. При этом выполняются аксиомы отношения эквивалентности:

1. Рефликсивность: $a = a$
2. Симметричность: $a = b \iff b = a$
3. Транзитивность: $a = b \wedge b = c \Rightarrow a = c$
4. Согласованность с функциональными символами: если $x_1 = y_1$, $x_2 = y_2$, ..., $x_k = y_k$, то $f(x_1, \ldots, x_k) = f(y_1, \ldots, y_k)$.
5. Согласованность с предикатными символами: если $x_1 = y_1, x_2 = y_2, \ldots, x_k = y_k$, то $p(x_1, \ldots, x_k) \Rightarrow p(y_1, \ldots, y_k)$. Обратная стрелочка получается автоматически из симметричности.

Теорема 2.3.1 (о полноте для нормальной модели)**.** Теория T с предикатом = имеет нормальную модель $\iff$ T непротиворечива, если к ней добавить все аксиомы равенства.

▶ **$\Rightarrow$:** Если есть нормальная модель, то в ней аксиомы выполняются, то есть, T с добавленными аксиомами совместна $\iff$ непротиворечива.

$\Leftarrow$: Если непротиворечива, то есть модель M. В ней = является некоторым отношением эквивалентности (возможно, «склеивающим» некоторые элементы и не подходящим под определение нормальной модели). Тогда определим новую интерпретацию: $M/=$ (фактор по отношению эквивалентности, функции и предикаты получится однозначно определить из аксиом). Эта модель получится нормальной моделью.

◀

Следствие 2.3.1.1 (теорема о компакности для нормальных моделей). Пусть T — теория с равенством. Если любая конечная подтеория имеет нормальную модель, то и T имеет нормальную модель.

▶ Упражнение, оно должно в точности повторять доказательство предыдущей теорему о компактности. ◀

Теорема 2.3.2 (Лёвенгейма-Сколема)**.** Если сигнатура теории не более чем счётна, то эта теория имеет не более чем счётную модель.

▶ Следует из доказательства теоремы о полноте: всякие там множества термов не более чем счётны (так как сигнатура не более чем счётна) и так далее, явно построим модель. ◀

Замечание 2.3.1. Мы знаем, что множество вещественных чисел несчётно. Однако же по предыдущей теореме у нас теория вещественных чисел имеет не более чем счётную модель.

Разрешение парадокса: когда мы доказывали несчётность вещественных чисел, мы жили «внутри» теории множеств (скажем, ZFC) и там доказывали всякое. Однако у ZFC есть счётная модель, а что мы там внутри доказываем — неважно.

То есть «счётное»/«несчётное» у нас двух видов.

Это надо знать на пятёрку на экзамене.

Утверждение 2.3.1. Пусть $Th(\mathcal{N}, 0, 1, +, \times, =)$ — множество замкнутых формул, истинных в арифметике.

Существует счётная модель этой теории, неизоморфная $\mathcal{N}$.

▶ Давайте допишем новую константу: c, допишем счётное число утверждений вида $c \neq 0$, $c \neq 1$, $c \neq 2$, ... Любое конечное подмножество новой теории имеет нормальную модель, так как мы из новых утверждений возьмём лишь конечное число утверждений, т.е. можно подобрать значение $c \in \mathcal{N}$. Значит по теореме о компактности есть какая-то нормальная модель для всей теории.

Однако носитель S этой модели не может быть изоморфен натуральным числам. В самом деле: если изоморфен, то c изоморфно какому-то числу $a \in \mathcal{N}$, но такого быть не может: у нас, в частности, есть условие $c \neq a$. ◀

Def 2.3.3. Теория T *разрешима*, если выводимость любой замкнутой формулы φ алгоритмически разрешима, то есть множество $\{\varphi \mid T \vdash \varphi\}$ разрешимо.

Def 2.3.4. Пусть σ — сигнатура, T — теория, φ — замкнутая формула. Тогда φ является *теоремой*, если $T \vdash \varphi$.

Def 2.3.5. Если Γ — множество теорем T. Тогда Γ — *аксиоматизация* T, если:

$$\forall \psi \in T \colon \Gamma \vdash \psi$$

То есть мы выводим какое-то другое множество теорем, из которого можно вывести исходные аксиомы $\Rightarrow$ можно вывести всё множество T и ничего больше.

Теорема 2.3.3. Пусть T полная, непротиворечивая и конечно аксиоматизируема (т.е. есть конечное множество аксиом). Тогда T разрешимо.

▶ Так как T полная, то из T можно вывести либо φ, либо $\neg\varphi$. Давайте запустим два алгоритма: один будет пытаться вывести φ, а другой — вывести $\neg\varphi$ (просто перебирая все строки-доказательства и проверяя). Один из них обязательно завершится. Получили требуемый алгоритм. ◀

Упражнение 2.3.1. Для перечислимого множества аксиом тоже верно.

Следствие 2.3.3.1. Множество $Th(\mathcal{N}, =, \times, +, 0, 1)$ полное. Однако оно неперечисислимо (по теореме Гёделя) $\Rightarrow$ неразрешимо $\Rightarrow$ нет перечислимой аксиоматизации.

Замечание 2.3.2. Иногда натуральные числа определяют через аксиомы Пеано в сигнатуре $\{=, +, \times, 0, 1\}$:

1. Аксиомы равенства
2. Для сложения и умножения (ассоциативность, дистрибутивность), Коммутативность — это сложная теорема.
3. Связь сложения и умножения
4. Аксиомы индукции (коих бесконечно): для любой формулы φ с одним параметром:

$$\varphi(0) \wedge (\forall k\colon \varphi(n) \to \varphi(n+1)) \to (\forall x\colon \varphi(x))$$

Натуральные числа являются моделью этих аксиом. Вообще, конечно, чтобы говорить «модель», нам нужна непротиворечивая теория. Непротиворечива ли аксиоматика Пеано — вопрос философский: мы можем лишь вывести это из веры в непротиворечивость какой-нибудь другой теории (например, Цермелло-Френкеля). Бывает даже объяснение, почему нельзя ничего поточнее.

Отсюда следует, что аксиоматика Пеано задаёт не все верные утверждения (потому что у всех верных не бывает перечислимой аксиоматизации, а аксиомы Пеано даже разрешимы). Об этом же, кстати, говорила первая теорема Гёделя о неполноте.

Теорема 2.3.4 (вторая теорема Гёделя о неполноте)**.** В арифметике Пеано невыразима формула C, которая выражает непротиворечивость арифметики Пеано. Без доказательства, это не самая строгая формулировка.

2.3.1. Примеры теорий

Целые числа

Знаем, что теория $Th(\mathbb{Z}, =, S, 0)$ разрешима (так как есть элиминация кванторов). Когда мы доказывали существование элиминации кванторов, мы пользовались какими-то свойствами модели с носителем $\mathbb{Z}$. Можно посмотреть на то, чем мы пользовались, и построить список необходимых аксиом:

1. $\forall x\colon \exists y\colon S(y) = x$ (S сюръективна)
2. $\forall x\colon \forall y\colon S(x) = S(y) \Rightarrow x = y$ (S инъективна)
3. $\forall x\colon S(x) \neq x$
4. $\forall x\colon S(S(x)) \neq x$
5. $\cdots$

6. $\forall x\colon S(\dots S(x)\dots) \neq x$

Оказывается, что этого достаточно для существования элиминации кванторов.

Упражнение 2.3.2. Проверить.

Теорема 2.3.5. У $Th(\mathbb{Z}, =, S, 0)$ нет конечной аксиоматизации.

▶ Пусть имеется конечная аксиоматизация $A_1, \dots, A_k$. Выведем каждую из A_i из исходных аксиом. В каждом выводе используется конечное число исходных аксиом — X_i. Объединим все X_i, получим лишь конечное число исходных аксиом, достаточных для вывода любой новой аксиомы $A_1, \dots, A_k$. Теперь возьмём минимальный исходных аксиом префикс, содержащий все исходные аксиомы из $\bigcup X_i$, назовём его L.

Заметим, что L тоже является аксиоматизацией теории (так как из него можно вывести аксиоматизацию A_i).

Пусть в L входят все исходные аксиомы $S^{(n)}(x) \neq x$, а вот аксиому $S^{(n+1)}(x) \neq x$ не взяли. Покажем, что она невыводима из предыдущих. В самом деле: возьмём модель $\mathbb{Z}/(n+1)\mathbb{Z}$ (вычеты по модулю $n+1$), в ней получаем $S^{(n+1)}(x) = x$, а все предыдущие аксиомы верны. ◀

Плотный линейный порядок

Def 2.3.6. Теория плотного линейного порядка T без первого и последнего элемента: есть сигнатура $\mathcal{P} = \{=, \leqslant\}$ (без функциональных символов) и следующие аксиомы:

1. Аксиомы равенства
2. $\forall x, y\colon (x \leqslant y) \vee (y \leqslant x)$
3. $\forall x\colon x \leqslant x$
4. $\forall x, y, z\colon (x \leqslant y) \wedge (y \leqslant z) \to x \leqslant z$
5. $\forall x, y\colon (x \leqslant y) \wedge (y \leqslant x) \to x = y$
6. $\forall x\colon \exists y\colon x < y$ (тут за $x < y$ обозначали $(x \leqslant y) \wedge \neg(x = y)$
7. $\forall x\colon \exists y\colon y < x$
8. $\forall x, y\colon ((x < y) \to (\exists z\colon (x < z) \wedge (z < y)))$ — плотность порядка

Замечание 2.3.3. Интерпретация с носителем $\mathbb{Q}$ является моделью T, так как все аксиомы верны. Отсюда ясно, что $Th(\mathbb{Q}, =, \leqslant)$ является надмножеством T, так как все аксиомы оно точно выполняет (пока непонятно, почему в него не попало что-то еще).

Утверждение 2.3.2. Для T нет конечных моделей.

▶ У нас для каждого элемента есть строго больший, причём отношение «строго больше» замкнуто по циклу быть не может. ◀

Утверждение 2.3.3. Все счётные модели T изоморфны.

▶ Пусть есть две счётные модели M_1 и M_2 с элементами $a_1, \dots$ и $b_1, \dots$. Берём по очереди по одному элементу из каждой модели: $a_1, b_1, a_2, b_2, \dots$. Взяли элемент, хотим найти ему соответствие в другой модели так, чтобы порядок был одинаков. Просто посмотрим, где в текущей модели стоит текущий элемент (между x и y) и воткнём его в другой модели между образами x и y.

Каждый элемент получит пару когда-нибудь, построили биекцию. ◀

Утверждение 2.3.4. T — полная теория.

Замечание 2.3.4. Если это так, то множество теорем теории T покрывает все верные утверждения, то есть совпадает с $Th(\mathbb{Q}, =, \leqslant)$.

▶ От противного: пусть T не является полной. Тогда возьмём какую-нибудь невыводимую формулу φ:

$$\begin{cases} T, \varphi & \text{непротиворечива} \\ T, \neg\varphi & \text{непротиворечива} \end{cases}$$

Рассмотрим не более чем счётные модели для каждой из двух нвоых теорий: M_1, M_2 (была теорема Лёвенгейма-Сколема). Они обе также являются моделями для T. Значит, они не могут быть конечными, т.е. они счётны. Следовательно, по предыдущему утверждению, они изоморфны.

Получаем противоречие: модели изоморфны, но в одной утверждение φ верно, а в другой — ложно. Так не бывает — это следует просто из того, как мы считаем истинность формул. ◀

Замечание 2.3.5. Наша теория T конечно аксиоматизирована аксиомами $A_1, \dots, A_k$. Это хорошо, так как проверка выводимости формулы φ сводится к проверке общезначимости формулы такого вида:

$$(A_1 \wedge A_2 \wedge \dots \wedge A_k) \to \varphi$$

Глава 3

Упорядоченные множества

Def 3.0.7. Есть множество M и бинарное отношение $\leqslant$. Оно называется *отношением частичного порядка*, если верны три свойства:

1. Рефлексивность: $a \leqslant a$
2. Транзитивность: $a \leqslant b \wedge b \leqslant c \Rightarrow a \leqslant c$
3. Антисимметричность: $a \leqslant b \wedge b \leqslant a \Rightarrow a = b$

Такое множество называется *частично упорядоченным множеством* (Ч.У.М.).

Def 3.0.8. Если $(M, \leqslant)$ обладает свойством:

$$\forall x, y \colon (x \leqslant y) \vee (y \leqslant x)$$

То множество называется *линейно упорядоченным*

Def 3.0.9. Пусть есть $(M_1, \leqslant)$ и $(M_2, \leqslant)$, то можно ввести ЧУМ $M_1 + M_2$: нарисовали M_1 слева, M_2 справа (две совсем независимые копии). При сравнении элементов из одного множества порядок старый (частичный), а из разного — элемент из M_1 всегда меньше любого элемента M_2.

Def 3.0.10. Можно ввести частичный порядок на декартовом произведении $M_1 \times M_2$:

$$(x_1, y_1) < (x_2, y_2) \iff (x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 < y_2)$$

Например, если x_1 и x_2 не сравнимы и не равны, то (x_1, y_1) и (x_2, y_2) тоже не сравнимы.

Напоминание:

Def 3.0.11. Элемент y минимальный, если:

$$\nexists x \colon x < y$$

Def 3.0.12. Элемент y наименьший, если:

$$\forall x \colon y \leqslant x$$

Def 3.0.13. M — Ч.У.М., оно называется *фундированным*, если любое его непустое подмножество имеет минимальный элемент.

Теорема 3.0.6. Следующие утверждения эквивалентны:

1. M — фундированное

2. В M нет бесконечно убывающей цепи ($x_1 > x_2 > x_3 > ...$)

3. Есть индукция: для любого предиката $A(x)$ над M верно:

$$(\forall x\colon$$
$$(\forall y\colon (y < x) \to A(y)$$
$$) \to A(x)$$
$$) \to \forall x\colon A(x)$$

Отдельное утверждение для базы не нужно: например, если у нас $M = \mathcal{N}$ (см. пример 3.0.1), то элемент $x = 0$ называется минимальным и левая часть при $x = 0$ вырождается в $\texttt{true} \to A(x)$.

▶ $1 \Rightarrow 2$**:** Пусть есть бесконечная цепь $x_1 > x_2 > ...$. Рассмотрим множество элементов этой цепи. Так как множество фундированное, то есть минимальный элемент x_k. Тогда неравенство $x_k > x_{k+1}$ не может быть верным.

$2 \Rightarrow 1$**:** Возьмём какое-нибудь непустое множество S. Возьмём какой-нибудь элемент x_1. Дальше возьмём какой-нибудь элемент $x_2 < x_1$. Потом — $x_3 < x_2$, и так далее. Если в какой-то момент не сможем найти нужный элемент, то последний взятый и есть минимальный. Иначе построим бесконечную убывающую цепь.

$1 \Rightarrow 3$**:** Пусть $S = \{x \mid A(x) —$ ложно$\}$. Пусть $y —$ минимальный элемент S. Тогда для любого $z < y$ верно, что $A(z)$ истинно. Следовательно, $A(y)$ истинно, т.е. $y \notin S$, противоречие.

$3 \Rightarrow 1$**:** Возьмём множество S, в котором нет минимального элемента. Возьмём характеристическую функцию $\chi_s(x)$ и предикат $P(x) \iff \chi_s(x) = 0$ (то есть предикат «$x \notin S$»). Покажем, что верно условие индукции (из пункта 3).

Зафиксировали некоторый x. Пусть для всех меньших y предикат уже верен, т.е. $y < x \Rightarrow y \notin S$. Тогда если $x \in S$, то x является минимальным элементом (так как все меньшие в S не лежат). Значит, $x \notin S$.

Тогда применяем принцип индукции, получаем, что $\forall x\colon P(x)$, то есть $S = \emptyset$.

◀

Пример 3.0.1. $\mathcal{N}$ фундированное.

Пример 3.0.2. Если A и B фундированные, то $A \times B$ тоже финдированно.

▶ Тут проще всего проверять отсутствие бесконечно убывающих цепей:

$$(a_1, b_1) > (a_2, b_2) > ...$$

Посмотрим на первые элементы. Из них нельзя выбрать бесконечную убывающую цепь, то есть они рано или поздно стабилизируются. Теперь посмотрим на вторые элемента, начиная с момента стабилизации: они тоже рано или поздно стабилизируются. ◀

Пример 3.0.3. $\mathcal{N}^k$ тоже фундированное ($\mathcal{N}^k = (\mathcal{N} \times \mathcal{N}) \times \mathcal{N} ...$).

Пример 3.0.4. Есть игра с бесконечным числом ходов: у нас бывают купюры k достоинств. На первом шаге есть какие-то. На каждом шаге мы отдаём любую одну купюру и берём любое конечное количество любых купюр меньшего достоинства.

Тогда в конце концов игрок останемся без денег.

▶ ◀

Def 3.0.14. $(M, \leqslant)$ называется вполне упорядоченным, если оно является фундированным и порядок линеен (в частности, минимальный и наименьших элементы — одно и то же).

Свойства в.у.м. M:

1. Имеется наименьший элемент, мы его будем обозначать 0 (ноль).

   ▶ Возьмём подмножество M, в нём есть минимальный элемент, он же — наименьший. ◀

2. Для каждого элемента $x \in M$, не являющегося максимальным, есть непосредственно следующий, то есть минимальный элемент, больший данного: $\min\{y \in M \mid y > x\}$. Давайте его обозначать $x + 1$.

   Пример 3.0.5. Возьмём бесконечное множество $\mathcal{N} + \{1, 2, \dots, k\}$. Тут есть максимальный элемент.

3. Некоторые элементы могут не иметь непосредственного предшественника.

   Def 3.0.15. Таким элементы называются *предельными*.

   Пример 3.0.6. В $\mathcal{N} + \mathcal{N}$ не имеют предшественника 0 и его копия из второго множества.

   Замечание 3.0.6. Из-за линейности порядка непосредственный предшественник не более, чем один.

4. Любой элемент в.у.м. можно представить в виде $z + n$, где z — некоторый предельный, а n — натуральное число, то есть в виде $z \underbrace{+1 + 1 + 1 + \dots + 1}_{n \text{ штук}}$.

   ▶ Возьмём элемент x и начнём брать непосредственного предшественника. Либо упрёмся в предельный через конечное число шагов, либо получим бесконечную цепь. ◀

5. Если $S \subseteq M$ и есть такой $y \in M$, что:

   $$\forall x \in S \colon x \leqslant y$$

   То существует точная верхняя грань S: это наименьший из множества верхних границ S.

Def 3.0.16. Пусть M — в.у.м. Тогда A называется *начальным отрезком*, если:

$$\forall x \in A, y \in M \setminus A \colon y > x$$

Свойства начальных отрезков:

1. Начальный отрезок в.у.м. сам является в.у.м.

2. Начальный отрезок начального отрезка — начальный отрезок исходного в.у.м.

3. Объединение любого семейства начальных отрезков — начальный отрезок.

4.

   Def 3.0.17. $[0, x) \overset{\text{Def}}{=} \{y \in M \mid y < x\}$

   $[0, x] \overset{\text{Def}}{=} \{y \in M \mid y \leqslant x\}$

   Эти два множества являются начальными отрезками.

5. Для любого начального отрезка A, не совпадающего со всем в.у.м., существует x такой, что отрезок совпадает с $[0, x)$.

   ▶ Возьмём $x = \min\{y \notin A\}$ ◀

6. Возьмём любые два начальных отрезка A и B. Тогда один целиком лежит в другом.

3.1. Трансфинитная индукция

Сейчас будем вести индукцию по произвольному вполне упорядоченному множеству. Например:

Теорема 3.1.1. Пусть A — в.у.м. и $f\colon A \to A$ — строго монотонно возрастающая функция ($x < y \Rightarrow f(x) < f(y)$). Тогда $f(x) \geqslant x$ для всех x.

▶ Докажем индукционных переход для элемента y. Мы предположили, что $\forall y < x\colon f(y) > y$. Тогда покажем, что $f(x) \geqslant x$ от противного: пусть $f(x) < x$. По индукционному предположению имеем $f(f(x)) \geqslant f(x)$. С другой стороны, по монотонности имеем $f(f(x)) < f(x)$. Противоречие. ◀

Теорема 3.1.2 (о трансфинитной рекурсии)**.** Пусть A — в.у.м., B — любое множество. Пусть F — некоторая функция, которая по элементу $x \in A$ и фукции $g\colon [0, x) \to B$ выдаёт элемент B. Тогда существует единственная $f\colon A \to B$ такая, что:

$$\forall x \in A\colon f(x) = F(x, f\,|_{[0;x)})$$

Замечание 3.1.1. Индукция — это доказательство утверждений, а рекурсия — вычисление значений функции, зная значения на предыдущих элементах.

▶ Будем говорить, что функция f *корректна* на $[0, a]$, если:

$$\forall c \leqslant a\colon f(c) = F(c, f\,|_{[0,c)})$$

Трансфинитной индукцией по a будем доказывать, что существует единственная корректная функция на $[0, a]$. Заодно поймём, что они согласованы: если f_{c_1} корректна на $[0; c_1]$, а f_{c_2} корректная на $[0; c_2]$, а также $c_1 < c_2$, то $f_{c_2}\,|_{[0;c_1]} = f_{c_1}$ (следует из единственности). Тогда можно будет определить искомую f так:

$$f(x) = f_x(x)$$

(берём единственную f_x, корректную на $[0; x]$ и берём её значение в x).

Итак, доказываем. Пусть доказали существование единственных корректных для всех меньших a, хотим показать единственную корректную f_a на $[0; a]$. Давайте подберём $h\colon [0, a] \to B$, согласованную со всеми корректными f_c ($c < a$):

$$h(x) = f_x(x), \text{где } f_x \text{ корректна на } [0; x]$$
$$h(a) = F(a, h\,|_{[0;a)})$$

Значения в предыдущих точках определены однозначно. Значит, h на $[0; a)$ определена однозначно. То есть $h(a)$ тоже однозначно из-за условия с F. ◀

Теорема 3.1.3 (о трансфинитной рекурсии для частично определённых функций)**.** Пусть A — в.у.м., B — любое множество. Пусть F — некоторая *частичная* функция (которая определена не везде), которая по элементу $x \in A$ и фукции $g\colon [0, x) \to B$ выдаёт элемент B. Тогда ровно одно из двух:

1. Существует единственная всюду определённая $f\colon A \to B$ такая, что:
$$\forall x \in A\colon f(x) = F(x, f\,|_{[0;x)})$$
2. Существует единственное $a \in A$ такое, что есть единственная всюду определённая $f\colon [0; a) \to B$ такая, что:
$$\forall x < a\colon f(x) = F(x, f\,|_{[0;x)})$$
а функция $F(a, f\,|_{[0;a)})$ не определена.

▶ Положим $B' = B \cup \{\bot\}$ ($\bot$ — «неопределённость»). Скажем, что F теперь определена везде, а где не была определена раньше — выдаёт $\bot$. Скажем, что если второй параметр F где-то принимает $\bot$, то результат — тоже $\bot$. Теперь пользуемся старой теоремой.

Замечание 3.1.2. Тут, наверное, надо сказать, что введение такого специального значения $\bot$ равносильно обычной неопределённости.

Если результатирующая f везде определена, то мы получили вариант 1. А если где-то неопределена, то рассмотрим минимальное место, где она неопределена — точку a. ◀

Теорема 3.1.4. Пусть A и B — вполне упорядоченные множества. Тогда одно из них изоморфно некоторому начальному отрезку другого.

Замечание 3.1.3. Из этого, в частности, следует, что любые два вполне упорядоченных множества можно сравнить по мощности (если одно изоморфно подмножеству другого, то его мощность не больше).

Потом докажем теорему Цермелло, которая гласит, что любое множество можно вполне упорядочить. Отсюда будет следовать, что любые два множества можно сравнить по мощности.

▶ Взяли два множества: A и B. Хочется взять наименьший элемент и там, и там, сопоставить, потом выкинуть, взять следующий наименьший, «и так далее». Хочется как-то это формализовать.

Будем определять отображение $f\colon A \to B$ по принципу трансфинитной рекурсии. Введём правило для трансфинитной рекурсии: $f(a)$ — наименьший элемент B, который не встречается в $\{f(x) \mid x < a\}$. Это не определено только в том случае, если $\{f(x) \mid x < a\} = B$. По правилу трансфинитной рекурсии у нас есть однозначная функция f (возможно, не везде определённая) и два случая:

1. f определена на всём A. Тогда хотим понять, что $f(A)$ — это начальный отрезок, причём изоморфный A.
   Покажем, что f — строго возрастающая функция. Возьмём две точки: a и b (пусть $a < b$). Заметим, что $f(a) \neq f(b)$, так как на шаге b мы выбираем значение, точное не совпадающее с предыдущими значениями. Также заметим, что $f(a)$ не может быть больше $f(b)$, иначе бы на шаге a мы бы выбрали $f(b) < f(a)$ в качестве минимального элемента.
   Теперь пишем условие того, что $f(A)$ — начальный отрезок:
   $$\forall b \notin f(A)\colon \forall c \in f(A)\colon c < b$$
   В самом деле: если $c > b$, то мы бы на шаге $f^{-1}(c)$ выбрали бы элемент b (он вообще ничьим образом не является) вместо c.
   Таким образом получили изоморфизм A и начального отрезка B.
2. Пусть f определена на начальном отрезке A, не совпадающем с A: $f\colon [0, a) \to B$. Тогда ясно, что $f([0, a)) = B$, так как мы не смогли найти значение в точке a.
   Аналогично предыдущему случаю f возрастает. То есть тоже получили изоморфизм B и начального отрезка A.

◀

Лемма 3.1.1. Никакое вполне упорядоченное множество не изоморфно своему собственному начальному отрезку (не совпадающему со всем множеством).

▶ В самом деле: пусть A вполне упорядочено, $[0, a)$ — его начальный отрезок ($a \in A$) и есть изоморфизм $f\colon A \to [0, a)$. Тогда f строго возрастает. А для возрастающей функции мы знаем, что $f(a) \geqslant a \notin [0, a)$, противоречие. ◀

Следствие 3.1.4.1. Для двух вполне упорядоченных множеств A и B всегда выполняется ровно один из трёх вариантов:

1. A изоморфно собственному начальному отрезку B
2. B изоморфно собственному начальному отрезку A
3. A изоморфно B

▶ Один из трёх выполняться должен (по теореме). Покажем, что не могут выполняться два сразу:

- Пусть A изоморфно собственному начальному отрезку $[0, b) \subsetneq B$ (есть биекция f), а B изоморфно собственному начальному отрезку $[0, a) \subsetneq A$ (есть биекция g).
  Тогда рассмотрим $g(f(A))$, это собственный начальный отрезок A ($g(f(A)) = g([0, b)) \subset [0, a)$).
  Т.о. A изоморфно собственному начальному отрезку, противоречие.
- Другая комбинации: пусть A изоморфно собственному начальному отрезку B, а B изоморфно A, тогда B изоморфно собственному начальному отрезку B, противоречие.

◀

Def 3.1.1. Аксиома выбора: пусть есть произвольное (в том числе несчётное) семейство непустых множество A_i, где $i \in I$. Тогда существует функция $f \colon I \to \bigcup A_i$ такая, что:

$$\forall i \in I \colon f(i) \in A_i$$

Следствие 3.1.4.2. Пусть A — множество. Давайте рассмотрим семейство $I = 2^A \setminus \{A\}$. Тогда по аксиоме выбора есть функция φ из этого семейства в A такая, что:

$$\forall x \subseteq A \colon \varphi(x) \in A \setminus x$$

В самом деле: положим $A_i = A \setminus i$.

Теорема 3.1.5 (Цермелло)**.** Всякое множество может быть вполне упорядочено.

Замечание 3.1.4. Эта теорема эквивалента аксиоме выбора (без доказательства). Аксиома выбора не слишком конструктивна, поэтому вряд ли для теоремы есть конструктивное доказательство.

▶ Возьмём некоторую фиксированную функцию φ из следствия к аксимое выбора.

Идея доказательства: возьмём элемент $\varphi(\emptyset)$ и называем его «минимальным». Следующий элемент (единицу) возьмём равным $\varphi(\{0\})$. Хочется сказать «и так далее», но надо обосновать.

Def 3.1.2. Назовём $(S, \leqslant_S)$ (где $S \subseteq A$) корректным фрагментом, если S — вполне упорядоченное множество (с порядком $\leqslant_S$), а также для любого $a \in S$:

$$a = \varphi([0, a))$$

Лемма 3.1.2. Если есть два корректных фрагмента $(S_1, \leqslant_{S_1})$ и $(S_2, \leqslant_{S_2})$, то один является начальным отрезком другого (возможно, несобственным; например $S_1 \subseteq S_2$, причём $S_1 = [0, s_2)$ для некоторого $s_2 \in S_2$) и на этом начальном отрезке порядки согласованы.

▶ Так как оба фрагмента являются в.у.м., то какое-то изоморфно начальному отрезку другого. Пусть $h \colon S_1 \to S_2$ — изоморфизм, причём S_1 изоморфно начальному отрезку S_2. Трансфинитной индукцией доказываем, что $h(x) = x$ (мы же помним, что $S_1, S_2 \subseteq A$, правда?).

Зафиксируем некоторый $x \in S_1$, пусть при $y < x$ верно $h(y) = y$. Надо проверить, что $h(x) = x$. Рассмотрим начальный отрезок $[0, x)_{S_1}$. По изоморфности $h([0, x)_{S_1}) = [0, h(x))_{S_2}$. По предположению индукции $[0, x)$ и $[0, h(x))$ совпадают как множества.

Так как S_1 и S_2 — коррекные фрагменты, из их определения можно написать:

$$x = \varphi([0, x)_{S_1})$$
$$h(x) = \varphi([0, h(x))_{S_2})$$
$$[0, x) = [0, h(x))$$
$$\Rightarrow \varphi([0, x)) = \varphi([0, h(x))$$
$$\Rightarrow x = h(x)$$

◀

Рассмотрим все корректные фрагменты: $(S_\alpha, \leqslant_{S_\alpha})$. Рассмотрим $S = \bigcup S_\alpha$. Введём порядок $\leqslant_S$ так: пусть $x, y \in S$, тогда $x \in S_1$, $y \in S_2$, тогда мы знаем, что либо S_1 изоморфно началу S_2, либо наоборот; откуда следует, что $x, y \in S_\alpha$ (для некоторого α). Тогда можно определить $x \leqslant_S y \iff x \leqslant_{S_\alpha} y$.

Утверждение 3.1.1. S — вполне упорядоченное множество.

▶ Оно точно с линейным порядком (любые два сравнимы), осталось проверить отсутствие убывающих цепей.

От противного: пусть есть убывающая цепь: $x_0 > x_1 > x_2 > \ldots$. Мы знаем, что $x_0 \in S_\alpha$ (для некоторого α). Покажем, что все $x_i \in S_\alpha$: рассмотрим какой-нибудь x_i, пусть он лежит в S_β. По лемме есть два случая:

- S_β является начальным отрезком S_α, тогда $S_\beta \subseteq S_\alpha$. Тогда, очевидно, $x_i \in S_\alpha$.
- S_α является начальным отрезком S_β: $S_\alpha = [0, c)_{S_\beta}$, то есть $x_i < x_0 < c$. Тогда $x_i < c$, т.е. $x_i \in S_\alpha$.

Тогда получаем, что у нас все элементы убывающей цепи лежат в S_α, т.е. нашли в S_α бесконечную убывающую цепь, противоречие. ◀

Также знаем, что для S верно:

$$\forall a \in S \colon a = \varphi([0, a))$$

Значит, S является корректным фрагментом.

Теперь надо понять, что $S = A$. От противного: пусть $S \subsetneq A$, и $a = \varphi(S)$ (помним, что тогда $a \notin S$) Тогда рассмотрим множество $S \cup \{a\}$, положим, что a строго больше всех элементов S. Тогда результат всё еще будет вполне упорядоченным множеством: порядок всё еще линеен, и бесконечных цепей возникнуть от добавления одного элемента не могло. Значит, $S \cup \{a\}$ — тоже корректный фрагмент, однако S равен объединению всех возможных корректных фрагментов, противоречие. ◀

Лемма 3.1.3 (Цорн)**.** Пусть Z — частично упорядоченное множество, а для любой цепи $A \subseteq Z$ (т.е. любые два элемента в A сравнимы) существует верхняя граница $b \in Z$ такой, что $\forall a \in A \colon a \leqslant b$.

Тогда для любого элемента $a \in Z$ существуют некоторый максимальный элемент m такой, что $m \geqslant a$ (максимальный — такой, что нет строго больших его).

Пример 3.1.1. В любом векторном пространстве (в том числе бесконечномерном) есть базис. Базис — это максимальное подмножество линейно независимых векторов.

▶ Пусть Z — множество, содержащее все наборы линейно независимых векторов. Операция порядка — просто $\subseteq$. Давайте проверим, что любая цепь имеет максимальный элемент. Взяли цепь A_i. Рассмотрим набор векторов $\bigcup A_i$, покажем, что он тоже линейно независим. В самом деле: если он линейно зависим, то некоторая конечная линейная комбинация векторов линейна зависима, каждый из векторов содержится в некотором элементе цепи, можно выбрать максимальный из элементов цепи A_i, получили, что этот элемент сам по себе уже линейно зависим.

Тогда по лемме Цорна получаем, что любое линейно независимое множество A можно вложить в некоторое максимальное линейно независимое множество A'. Тогда A' по определению является базисом. Мы даже доказали что-то более сильное. ◀

▶ Доказываем лемму Цорна: рассмотрим множество I — множество строго большей мощности, чем Z (например, 2^Z). По теореме Цермелло вполне упорядочим I. Доказываем лемму Цорна от противного: пусть для некоторого элемента $a \in Z$ нет максимального элемента, больше либо равного a (замечание: «больше либо равно» не есть «не меньше» в частично упорядоченном множестве). Теперь рассмотрим функцию $f\colon I \to Z$, положим $f(0) = a$, а остаток зададим трансфинитной рекурсией. Будем задавать $f(x)$. Положим $F = f\,|_{[0,x)}$.

- Если такое суженное f не строго возрастает, то говорим, что $f(x)$ неопределено
- Если суженное f строго возрастает, то F есть цепь, пусть s — верхняя граница этой цепи, причём $f(0) \in F$ и $a = f(0) \leqslant s$. Тогда s не может быть максимальным элементом, иначе $a \in s$. Тогда существует некоторый $t > s$, положим $f(x) = t$.

Два случая:

- Пусть f определена только на $[0, b)$. Тогда $f\,|_{[0,x)}$ немонотонна, т.е. в каких-то двух точках $c < d$ имеем $f(c) > f(d)$. Но мы тогда, получается, неверно определили $f(d)$, так как на сооветствующем шаге в F лежал элемент $f(c)$, и мы брали элемент, строго больший верхней границы, в частности, строго больший $f(c)$.
- Пусть f определена на I. Тогда мы построили монотонно возрастающее отображение (т.е. инъекцию) из I в Z (откуда $|I| \geqslant |Z|$), но $|I| < |Z|$ (по построению I), противоречие.

◀

Пример 3.1.2. Пополнение списка формул до полного непротиворечивого. Пусть Γ — непротиворечивый список пропозициональных формул, тогда существует $\Gamma' \supseteq \Gamma$ — полный непротиворечивый список (пополнение)

▶ Вводим частично упорядоченное множество, элементы которого — непротиворечивые формулы Γ с данным множеством переменных. Отношение — включение.

Давайте покажем, что у любой цепи Γ_i есть верхняя граница. В самом деле, $\bigcup \Gamma_i$ есть набор формул, причём непротиворечивый, потому что противоречие всегда выводится из конечного множества формул. А тогда это конечное множество формул целиком лежит в каком-то из Γ_i (потому что среди конечного числа элементов цепи Γ_i можно выбрать наибольший).

Тогда существует максимальный элемент $\Gamma' \supseteq \Gamma$. Покажем, что Γ' полон: в самом деле, если для некоторого φ нельзя вывести ни φ, ни $\neg\varphi$, то $\Gamma' \cup \{\varphi\}$ тоже непротиворечиво (было правило: если $\Gamma', \varphi \vdash A, \neg A$, то $\Gamma' \vdash \neg\varphi$).

Таким образом, Γ' полно. ◀

Для предикатных формул можно аналогично, но надо говорить про замкнутые формулы.

Пример 3.1.3. Если A бесконечно, то $|A \times A| = |A|$.

▶ Идея доказательства (деталей в этом доказательстве не будет): для некоторых подмножеств A это верно (в частности, для счётных). Назовём подмножество *хорошим*, если для него это верно. Далее будем рассматривать пары (подмножество, биекция) и введём порядок: подмножества вложены, а биекции согласованы.

Тогда можно рассмотреть максимальную пару (B, f), и что-то там показать. ◀