Знакомство с математической логикой и теорией алгоритмов: математика и философия теорем Геделя о неполноте

Гумин Максим

9 февраля 2011 г.

Аннотация

Первая теорема Геделя о неполноте является, пожалуй, самой известной теоремой математики XX века. Она утверждает, что при любом определении формального доказательства найдутся истинные, но не доказуемые утверждения. Согласно второй теореме о неполноте, непротиворечивость достаточно общих систем аксиом, типа системы аксиом ZF теории множеств, невозможно доказать финитными методами. В настоящем тексте приводятся строгие доказательства теорем о неполноте и обсуждается, какое именно препятствие они представляют для обоснования математики с позиции формализма. Предварительно вводятся все необходимые понятия математической логики и теории алгоритмов. Предпринята попытка философского осмысления языков первого порядка и феномена частичных решателей алгоритмически неразрешимых задач. С другой стороны, показано, какую роль в математике играют такие философские идеи как логический позитивизм, категориальная система Канта, нотация Рассела и проблема бороды Платона. Разбираются также типичные злоупотребления теоремами Геделя. Корректно ли, например, утверждать, что эти теоремы указывают на невозможность создания сильного ИИ? Впрочем, целью настоящего текста было не столько сказать что-либо новое, сколько помочь автору разобраться в теме.

Содержание

1 Введение **3**

1.1 Это предложение ложно 3

1.2 Истина и язык 4

1.3 Парадоксов нет! 7

1.4 Страшные числа 8

2 Чего не могут компьютеры **10**
2.1 Алгоритмическая неразрешимость 10
2.2 Уточнение понятия «алгоритм» 14
2.3 Проблема остановки 15
2.4 Как превзойти алгоритм 16
2.5 Другие алгоритмически неразрешимые проблемы 17
2.6 Разрешимые и перечислимые множества 19

3 Логика предикатов **21**
3.1 О том, что есть 21
3.2 Формальные языки первого порядка 24
3.3 Семантика языков первого порядка 26
3.4 Выразимость предикатов 28
3.5 Автоморфизмы интерпретаций и невыразимость 31
3.6 Выразимость в арифметике 32

4 Что такое доказательство? **35**
4.1 Формальные доказательства 35
4.2 Несколько метатеорем 40
4.3 Неформальные доказательства 43
4.4 Формализм как программа обоснования математики 44

5 Теоремы Геделя о неполноте **46**
5.1 Насколько хороша истина? 46
5.2 Теорема Тарского о невыразимости истины 47
5.3 Еще одно доказательство теоремы Геделя 49
5.4 Вторая теорема Геделя о неполноте 49

6 Злоупотребления теоремой Геделя о неполноте **51**
6.1 Стандартные злоупотребления 51
6.2 Искусственный интеллект и искусственное сознание 53
6.3 Компьютерное моделирование физики 55

7 Приложения **57**
7.1 Аксиоматика ZF теории множеств 57
7.2 Аксиоматика Тарского евклидовой геометрии 58
7.3 Аннотация для ЛМШ 59

1 Введение

1.1 Это предложение ложно

1. Истинно или ложно предложение

(1) «это предложение ложно»?

Правильный ответ в предыдущей задаче такой: задача некорректна! С чего мы взяли, что последовательность из нескольких букв русского алфавита должна быть истинна или ложна? Да и что такое истина? Никого же не удивит, что невозможно определить истинность предложения

(2) «поуГен лдводавл бьыдДДщшеоь здюююгй»

Конечно, это никого не удивит, потому что предложение (2) синтаксически неправильно: в русском языке нет слов «поуГен» и «лдводавл». Хорошо, а как тогда быть с предложением

(3) «зеленые идеи яростно спали»?

Это предложение уже синтаксически правильное: входящие в него слова есть слова русского языка, прилагательное «зеленые» согласовано с существительным «идеи» и т.д.. Но придать ему смысл и какое-то истинностное значение не получается, потому что идеи не могут быть зелеными, не могут спать и потому что нельзя спать яростно. Ладно, можно попытаться запретить применять к слову «идея» такие прилагательные как «зеленая» или «деревянная» и разрешить применять такие прилагательные как «революционная» или «чужая». Но сильно ли это поможет? Например, истинно или ложно предложение

(4) «белая кошка сидит на подоконнике»?

А какая именно кошка имеется в виду? Наверняка в данный момент времени хотя бы одна белая кошка сидит на каком-то подоконнике. Впрочем, она вряд ли идеально белая. Или речь шла только о подоконниках в этой комнате?

Можно, конечно, еще добавлять к предложениям контекст, проясняющий детали. Но, я думаю, из приведенных примеров уже понятно, что предложение совершенно не обязательно должно иметь какой-либо смысл или истинностное значение. Предложение — это всего лишь последовательность букв некоторого алфавита, удовлетворяющая некоторым синтаксическим правилам.

Однако здесь можно пойти слишком далеко: можно сказать, что смысл и истина — это иллюзии, и все, с чем мы имеем дело — это всего лишь набор букв. Но такой подход является противоречивым. Потому что если кто-то говорит, что истина — это иллюзия, то он с необходимостью должен считать предложение

(5) «истина — это иллюзия»

истинным. Что противоречит самому предложению (5). Поэтому давайте будем относиться к предложениям априори только как к последовательности букв, но помнить, что полностью отказываться от понятий смысла и истины нельзя. Если у нас получилось придать какой-то последовательности букв смысл, то нам очень повезло.

Эта же проблема есть основная проблема эмерджентного материализма. Согласно эмерджентному материализму, мир состоит из некоторых элементарных компонент (атомов, частиц, полей — неважно), движущихся и взаимодействующих согласно каким-то законам (возможно, вероятностным или неизвестным нам), которые в принципе могут быть проверены экспериментом. Другими словами, мир состоит только из частиц, удовлетворяющих законам, которые может проверить внешний по отношению к системе наблюдатель. Эти частицы способны порождать очень сложные структуры, в том числе ткани живых существ и человеческий мозг. Таким образом, работу мозга можно полностью свести к законам взаимодействия этих элементарных частиц. Все мысли людей — это всего лишь необычайно сложные процессы, происходящие в их мозге. Но на уровне элементарных частиц нет места истинности. Получается, что наше свойство считать некоторые предложения истинными — это всего лишь психологическая иллюзия. Проблема такого подхода заключается в том, что то же относится и к предложению

(6) «на самом деле верен эмерджентный материализм»

Получается, что эмерджентный материализм невозможно даже корректно сформулировать.

Вернемся к парадоксу лжеца. Мы только что выяснили, что предложение «это предложение ложно» не более противоречиво, чем предложение «поуГен лдводавл бьыдДДщшеоь здюююгй». Мир не кончится, если на листке бумаги написать «это предложение ложно». Почему же (1) кажется более парадоксальным, чем (2)? Дело в том, что, когда мы читаем (1), нам кажется, что ничто не мешает этому предложению иметь истинностное значение: оно синтаксически правильное, слова согласованы друг с другом и употреблены верно.

Совершенно отдельный и открытый вопрос — каким условиям должно удовлетворять предложение, чтобы ему гарантированно можно было придать истинностное значение? Пока можно сказать, что причиной бессмысленности некоторого предложения может быть его свойство ссылаться на себя же. Предложение (1), например, ссылается само на себя, что особенно заметно, если его переписать в виде

$$A = \text{«A ложно»}$$

1.2 Истина и язык

Если мы хотим использовать математику для понимания феноменов истинности и смысла, то нам нужно придумать некоторый формальный язык. Русский язык не формализован и слишком сложен для этой цели. Про математически формальный

язык можно будет даже попробовать доказать какие-нибудь теоремы! Полностью такой формальный язык мы опишем несколько позднее, а пока просто приведем примеры предложений на нем:

$$(A) \qquad «\forall n\big((2|n) \wedge (n \neq 2) \to \exists k \exists l (k \text{ — простое}) \wedge (l \text{ — простое}) \wedge (n = k + l)\big)»$$

$$(B) \qquad «\forall m \forall n (m + n = n + m)»$$

$$(C) \qquad «\exists n\Big(n > 10 \wedge \forall m \forall k\big((m-3)k = n\big)\Big)»$$

Это, конечно, были синтаксически правильные предложения. Пример синтаксически неправильного предложения привести тоже просто:

$$(D) \qquad « + n))\forall = m(\cdot \exists l m(((26»$$

Но! Удивительная вещь! Каждому синтаксически правильному (и замкнутому) предложению (X) на этом языке мы можем сопоставить некоторое истинностное значение! А именно, пусть «$\exists n$» означает «существует такое натуральное число n, что», «$\forall n$» означает «для любого натурального n выполняется следующее», $n+m$ означает сумму двух натуральных чисел n и m и т.д.. Тогда по предложению (X) мы получаем некоторое утверждение о натуральных числах. А оно либо истинно, либо ложно! Например, предложение (C) ложно, предложение (B) истинно, а предложение (A) либо истинно, либо ложно, но конкретно нам неизвестно, какой из этих двух случаев выполняется (предложение (A) называется гипотезой Гольдбаха).

Разберемся, в чем здесь отличие от русского языка. Конечно, на нашем формальном языке уже не сформулируешь что-нибудь типа «истина — это иллюзия» или «это предложение ложно»! Он очень беден по сравнению с русским. На нем формулируются разве что некоторые утверждения о натуральных числах. Но это еще не все! Например, предложение

$$(4) \qquad «\text{белая кошка сидит на подоконнике}»$$

не ссылается само на себя и говорит о совершенно не абстрактных вещах типа кошки и подоконника. Но придать ему истинностное значение не получается. Точнее, для того, чтобы сказать, истинно (4) или ложно, нам еще необходим человек, который как-то интерпретирует это предложение. Который будет знать, о какой именно белой кошке идет речь, чем отличается «сидеть» от «лежать» и какими подоконниками следует ограничиться. А предложения типа (A), (B), (C) истинны или ложны вне зависимости от кого бы то ни было! Или все-таки нет?

Для формулировки и доказательства теорем Геделя нам необходимо принять два «метафизических» предположения:

1. Предложениям настоящего текста (написанным на русском языке) можно придать какой-то смысл. Может быть, не всем. Может быть, разный для разных людей. Но для формулировки и доказательства теоремы Геделя нельзя относиться к настоящему тексту только как к конечной последовательности букв.

Чтобы не путать предложения рассматриваемого формального языка с предложениями метаязыка, мы постоянно будем использовать кавычки.

2. Предложениям типа (A), (B), (C) (т.е. замкнутым синтаксически правильным предложениям на формальном языке арифметики) однозначно соответствуют некоторые истинностные значения, причем это соответствие не зависит от конкретного человека. Договоримся считать, что натуральные числа существуют реально (в каком-то смысле) и одни и те же для всех людей. Договоримся, что не может быть такого, что для одного человека гипотеза Гольдбаха верна, для другого — нет, а для третьего у нее нет истинностного значения.

Другими словами, для формулировки теорем Геделя необходимо быть немного платонистом (хотя практически все ее доказательство может быть проведено, например, в аксиоматике Пеано — подробнее об этом дальше).

Итак, каждому синтаксически правильному предложению (X) на формальном языке соответствует некоторое истинностное значение $T(X)$, равное 0 или 1. Однако это соответствие какое-то туманное. Для определения $T(X)$ мы привлекаем какие-то метафизические соображения типа существования натуральных чисел. Нельзя ли сделать соответствие $(X) \mapsto T(X)$ более ясным? Попытки придать этому соответствию более ясный смысл, и «схватить», таким образом, понятие истинности синтаксическими средствами привели к следующим трем программам действий:

1. О реализации самой смелой из этих программ мечтали еще Декарт и Лейбниц в XVII веке. Давайте найдем некоторую универсальную процедуру, которая по предложению (X) будет говорить нам, истинно оно или ложно! Другими словами, давайте напишем компьютерную программу, которая на вход будет принимать последовательность символов (X), а на выходе возвращать $T(X)$. Если это получится, то математики больше будут не нужны (по крайней мере те, что занимаются определением истинности утверждений о натуральных числах).

2. Если все-таки алгоритм определения истинности по каким-то причинам найти не удастся, то можно поставить более достижимую цель. Как известно, истинность утверждений математики определяют с помощью доказательств. Давайте найдем такое формальное определение доказательства (или такую систему аксиом), для которой все истинные утверждения о натуральных числах могут быть доказаны!

3. Если же и предыдущая программа потерпит неудачу, то можно поступить еще лучше. Хорошо, все истинные предложения доказать не получается. Давайте тогда вообще откажемся от понятия истинности предложений о натуральных числах! Будем работать только с понятием доказуемости. Правда, возникает одна проблема: если наша система аксиом противоречива, то *все* утверждения будут в ней доказуемы. Не беда. Непротиворечивость системы аксиом мы тоже докажем, ведь это обычная комбинаторная задачка! Если это получится то про

понятие истинности можно будет забыть вообще, и оставить только понятие доказуемости!

Следуя идеям математиков XX века (Геделя, Тьюринга, Тарского и других), мы докажем (!), что каждая из этих программ обречена на провал.

1.3 Парадоксов нет!

2. В некотором царстве брадобрею приказали брить тех и только тех, кто не бреется сам. Есть ли в этой истории логическое противоречие?

Решение: очевидно нет, т.к. ничто не мешает отдать такой приказ. Очевидно также, что этот конкретный приказ исполнить невозможно (такие случаи — не редкость в реальных бюрократических системах). Действительно, если предположить, что приказ корректен, то каждый из двух случаев «брадобрей бреет себя сам» и «брадобрей не бреет себя сам» приводит к противоречию. Парадоксальность же ситуации заключается в том, что приказ кажется даже естественным: естественно приказать побрить именно тех, кто не бреется сам.

Парадокс брадобрея тесно связан с парадоксом Рассела наивной теории множеств. Рассел предлагает называть множество хорошим, если оно не содержит себя в качестве элемента. Рассмотрим тогда множество хороших множеств. Это множество хорошее или нет? Легко проверить, что любой вариант ответа на этот вопрос приводит к противоречию. Интересное отличие этого парадокса от парадокса брадобрея заключается в том, что здесь нет никаких «предложений», «сказал», «приказал» и т.д.. Точнее, синтаксические конструкции здесь используются неявно: не получается придать какой-либо смысл строке «множество хороших множеств». В отличие парадокса Брадобрея, из которого не вытекает никаких следствий, парадокс Рассела поставил крест на наивной теории множеств.

3. Некоторые предложения на русском языке описывают натуральные числа. Например: «миллион», «сто тридцать семь», «наименьшее натуральное число, представимое в виде суммы двух кубов двумя различными способами». Давайте ограничимся только предложениями, длина которых меньше 100 символов. Таких предложений конечное число, поэтому множество натуральных чисел, которые нельзя описать предложениями короче 100 символов, непусто. Рассмотрим тогда наименьшее натуральное число, которое нельзя описать предложением короче 100 символов. Но ведь мы только что его описали следующим предложением:

«наименьшее натуральное число, которое нельзя
описать предложением короче 100 символов»!

Объясните возникшее противоречие.

Решение: Для того, чтобы строго определить соответствие между натуральными числами и их описаниями, необходимо фиксировать какой-то конкретный формальный язык $\mathcal{L}$ (являющийся некоторым формализованным ограничением русского языка). Предложение

«наименьшее натуральное число, которое нельзя
описать предложением на языке $\mathcal{L}$ короче 100 символов»

не является предложением на языке $\mathcal{L}$. Поэтому никакого противоречия здесь нет.

4. Однажды учитель объявил ученикам, что на следующей неделе он проведет контрольную работу. При этом накануне контрольной они не будут знать точно, что контрольная завтра (уроки проходят каждый день с понедельника по пятницу). Ясно, что в пятницу контрольной быть не может: тогда в четверг вечером школьники точно будут знать, что контрольная завтра. Но тогда в среду вечером они точно будут знать, что контрольная в четверг, ведь в пятницу ее быть не может. Значит, и в четверг не может быть контрольной... Словом, ученики пришли к выводу, что контрольной вообще не будет, и успокоились. А контрольная произошла в среду. Объясните этот парадокс.

5. Племя людоедов поймало Робинзона Крузо. Вождь сказал: «Мы бы рады тебя отпустить, но по нашему закону ты должен произнести какое-нибудь предложение. Если оно окажется истинным, мы тебя съедим. Если оно окажется ложным, тебя съест наш лев». Что нужно сказать Робинзону, чтобы не быть съеденным?

Комментарий. Условие этой задачи можно понимать по-разному. Один вариант: Робинзону нужно произнести предложение, придать истинностное значение которому людоеды не смогут. В этом случае ему не обязательно говорить «меня съест лев». Можно сказать что-нибудь вроде «У!», или «зеленые идеи яростно спали», или даже «белая кошка сидит на подоконнике». Континуум-гипотеза бы тоже сработала. Другой вариант понимания условия: Робинзону нужно расширить систему аксиом аборигенов до противоречивой. О системах аксиом мы будем говорить позже.

1.4 Страшные числа

Цель настоящего раздела — помочь слушателям освоиться с понятием счетности.

6. Докажите, что множество конечных слов в конечном алфавите счетно.

Определение 1. *Число $x \in \mathbb{R}$ называется вычислимым, если существует алгоритм A, последовательно выводящий все цифры числа x после запятой.*

Это определение ссылается на неопределенное пока понятие алгоритма. Будем считать, что алгоритм — это всего лишь программа на каком-нибудь фиксированном языке программирования, например на Си.

Очевидно, все целые числа вычислимы, потому что алгоритму достаточно просто выводить нули. Например, следующий алгоритм выводит все цифры числа 19:

```
void main () {
   Write ( '1' );
   Write ( '9' );
   Write ( ',' );
   while ( true ) Write ( '0' );
}
```

Все рациональные числа тоже вычислимы, т.к. они соответствуют периодическим десятичным дробям, а периодические последовательности легко выводить используя ту же конструкцию `while`. Все алгебраические числа вычислимы, т.к. существуют методы, находящие действительные решения алгебраических уравнений со сколь угодно большой точностью. Число π тоже вычислимо, т.к. оно приближается рациональными числами вида

$$s_n = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + (-1)^{n+1}\frac{1}{2n-1}\right), \qquad |\pi - s_n| < \frac{4}{n}$$

Аналогично, вычислимо число e, потому что оно приближается такими конечными суммами:

$$s_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!}, \qquad |e - s_n| < \frac{1}{n!}$$

Вычислимы, например, и решения уравнений типа $x^2 = \sin x$ — для них тоже алгоритмы, дающие решения со сколь угодно большой наперед заданной точностью... Может быть, все действительные числа вычислимы?

7. Существует ли хоть одно невычислимое число?

Решение: вычислимых чисел не больше, чем программ на языке Си. А множество программ счетно (т.к. программа — это конечный текст в конечном алфавите). Поэтому множество вычислимых чисел счетно. Множество же всех действительных чисел несчетно. Следовательно, невычислимые числа существуют! Более того, их гораздо больше чем вычислимых! Но почему-то пример конкретного невычислимого числа привести не получается!

8. Число $x \in \mathbb{R}$ называется трансцендентным, если не существует ни одного многочлена P с рациональными коэффициентами, для которого $P(x) = 0$. Докажите, что существуют трансцендентные числа.

Кантор придумал доказательство существования трансцендентных чисел, основанное на соображениях мощности, хоть и раньше, чем была доказана трансцендентность чисел π и e, но позже, чем были известны конкретные примеры трансцендентных чисел. В то время его доказательство было воспринято как какой-то фокус.

9. Докажите, что множество $\mathbb{Q}$ рациональных чисел счетно.

Решение: $\mathbb{Q}$ является подмножеством множества конечных слов в алфавите

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, /\}$$

(точнее, $\mathbb{Q}$ туда вкладывается). А множество конечных слов в любом конечном алфавите счетно. Поэтому $\mathbb{Q}$ счетно.

2 Чего не могут компьютеры

2.1 Алгоритмическая неразрешимость

В школе вы решали квадратные уравнения. Как решить уравнение $x^2-5x+6=0$? Известно, как: нужно подставить числа $a=1, b=-5, c=6$ в формулу $\frac{1}{2a}(-b\pm\sqrt{b^2-4ac})$. А как решить уравнение $|x+2|+|x-1|=3$? Для таких задач тоже известен общий алгоритм: нужно рассматривать случаи $x<-2$, $-2\leqslant x<1$, $1\leqslant x$ и для каждого из этих случаев решать получающееся линейное уравнение. Во многих текстовых задачах, например, в задачах на смеси, тоже нет шансов не догадаться до решения, если вам известен алгоритм: по тексту задачи совершенно механически составляется система линейных уравнений, а системы линейных уже решаются известным алгоритмом.

Может быть, все задачи такие? Может быть, для каждого класса задач существует алгоритм, следуя которому можно решить любую задачу из этого класса? Опять же, если мы хотим ответить на этот вопрос точно, то необходимо его формализовать. Например, что имеется в виду под алгоритмом решения квадратных уравнений? Решения квадратного уравнения с целыми коэффициентами — это, вообще говоря, действительные числа. А компьютеры в принципе не умеют работать с бесконечными непериодическими последовательностями. Или имеется в виду приближенный алгоритм, возвращающий правильный ответ с произвольной наперед заданной точностью? Или ответ разрешается дать в виде строчки символов, среди которых может встретиться знак квадратного корня?

Пример неформализованного класса задач: «На вход подается текст задачи по планиметрии, написанный на русском языке. Программа должна решить эту задачу». Этот пример не годится сразу по нескольким причинам: во-первых, русский язык — это не формальный язык; во-вторых, без дополнительных уточнений не понятно, что такое «решить».

Рассмотрим несколько формальных классов задач, относительно которых не возникает сомнений в их алгоритмической разрешимости:

1. Сортировка массива целых чисел. Алгоритмическая разрешимость этого класса означает, что существует программа, принимающая конечные последовательности целых чисел и возвращающая последовательности, составленные из тех же чисел, но упорядоченных по возрастанию.

2. Поиск пути в графе. На вход программы подаются конечный граф (закодированный каким-то образом в виде строчки символов) и две его вершины, а вернуть она должна путь в графе, соединяющий две эти вершины.

3. Решение кубика Рубика. Вход: состояние кубика Рубика (или игры «15»). Выход: последовательность операций, переводящая кубик Рубика в правильное состояние.

4. Разложение на простые множители. Вход: натуральное число. Выход: разложение этого числа на простые множители.

5. Вычисление рода. Вход: конечный симплициальный комплекс. Программа должна определить, гомеоморфна ли геометрическая реализация этого комплекса двумерной ориентируемой поверхности, и в случае положительного ответа найти род этой поверхности.

6. Проверка синтаксической правильности. На вход программы подается предложение на формальном языке арифметики. Программа должна определить, является ли это предложение синтаксически правильным и замкнутым.

Как вы думаете, какая из этих задач самая простая? Конечно, существуют очень короткие алгоритмы для (1) и (4), но задача (3) принципиально проще остальных! Попробуем придумать соответствующий алгоритм. Первое, что приходит в голову, это свести (3) к (2): действительно, построим конечный граф, вершинами которого будут все возможные состояния кубика Рубика, и в котором две вершины соединены ребром, если и только если одно из этих состояний можно перевести в другое элементарным движением кубика. Ясно, что алгоритм, строящий такой граф, будет далеко не оптимальным в плане используемой памяти и времени работы, но нас сейчас не должны волновать подобные мелочи. Гораздо важнее сам факт существования такого алгоритма. Существуют гораздо более эффективные алгоритмы, использующие специфику кубика Рубика. Например, в игре «15», любую транспозицию фишек легко представить в виде композиции движений.

Но до сих пор мы думали как программисты, а не как математики. Заметим, что состояний у кубика Рубика лишь конечное число (хотя и очень большое). Значит, у программы может быть только конечное число различных входов! А для каждого из этих входов мы можем явно указать требуемую последовательность элементарных движений! Таким образом, программа, собирающая кубик Рубика, будет выглядеть так:

```
int main ( int input ) {
   if ( input == 1 ) return ...;
   if ( input == 2 ) return ...;
   ...
   if ( input == 43252003274489856000 ) return ...;
}
```

Перейдем к более интересным примерам. В 1900 году на Международном математическом конгрессе в Париже Давид Гильберт сформулировал 23 математические

проблемы, которые, на его взгляд, являлись наиболее важными для математики начинающегося XX столетия. Проблема под номером 10 звучала так: найти алгоритм, определяющий, имеет ли данное диофантово уравнение хотя бы одно целочисленное решение. Диофантово уравнение — это алгебраическое уравнение с целыми коэффициентами, с произвольным числом переменных, например

$$m^2 + n^2 + l^2 + 8k + 1 = 0$$

Попробуем сейчас построить такой алгоритм. Пусть диофантово уравнение на входе имеет N переменных. Множество

$$\mathbb{Z} \times \ldots \times \mathbb{Z}$$

(всего N множителей) счетно. Поэтому мы просто можем перебирать все возможные наборы из N целых чисел и подставлять их во входное уравнение. Если какой-то набор оказывается решением, то возвращаем единичку. Если набор — не решение, то перебираем дальше. Получилось, что мы решили проблему Гильберта. Где здесь ошибка?

Ошибка в том, что, если решение уравнения не находится, то не понятно, когда нужно останавливаться и возвращать 0. Другими словами, на уравнении, не имеющем решений, приведенный алгоритм впадает в бесконечный цикл и не останавливается. А нам бы хотелось, чтобы он когда-нибудь остановился и возвратил 0.

Итак, важная особенность алгоритмов — их свойство впадать иногда в бесконечный цикл и не заканчивать работу. Эта особенность лежит в основе определения вычислимой функции.

Определение 2. *Функция $f\colon \mathbb{N} \to \mathbb{N} \cup \{*\}$ называется вычислимой, если существует такая программа A, что для любого $n \in \mathbb{N}$ $f(n) = A(n)$, если A заканчивает работу на входе n, и $f(n) = *$, если A не заканчивает работу на входе n. Вычислимая функция, для которой не существует такого $n \in \mathbb{N}$, что $f(n) = *$, называется тотально вычислимой.*

Другими словами, вычислимая функция — это функция, которая может быть реализована некоторым алгоритмом, а тотально вычислимая функция — это функция, которая может быть реализована всегда останавливающимся алгоритмом. Очевидно, невычислимые функции существуют. Действительно, множество всех программ счетно, а множество всех функций $\mathbb{N} \to \mathbb{N}$ имеет мощность континуума. Но, как и с вычислимыми числами, конкретный пример невычислимой функции (с доказательством невычислимости) привести не получается. Скоро мы предъявим такой конкретный пример!

10. Существует ли такая программа:

> 7. На вход программы (будем называть ее базовой) подаются другие программы (которые будем называть входными). Можно считать, что входные

программы сами не имеют входов. От базовой программы требуется вернуть результат работы входной программы. Если же входная программа не останавливается, то и базовая программа не должна останавливаться.

Такие программы называются интерпретаторами (или универсальными машинами, или виртуальными машинами). Конечно, они существуют: никого не удивит, что можно написать интерпретатор языка Си на языке Си.

Следующий пример. Представьте, что вас попросили написать специальную (8) «программу-анализатор». На вход этого анализатора может быть подана любая программа без входов. От анализатора требуется определить, остановится входная программа или нет. Другими словами, определим функцию $f\colon \mathbb{N} \to \{0,1\}$ следующим образом: $f(n) = 0$, если программа с номером n останавливается, и $f(n) = 1$, если программа с номером n не останавливается. Является ли функция f вычислимой? В следующем разделе мы докажем, что ответ на этот вопрос отрицательный.

Напомним, что одна из основных целей курса — доказать, что функция (9), сопоставляющая каждому синтаксически правильному предложению X на формальном языке арифметики его истинностное значение $T(X)$, является невычислимой. Кратко мы можем переформулировать это так: арифметика неразрешима. И неформально продолжить: а раз уж арифметика неразрешима, то вся математика и подавно. Не существует алгоритма, решающего любую математическую задачу (даже точно формализованную). Не существует алгоритма, определяющего истинность любого утверждения. Не существует абсолютного алгоритмического ИИ: не какого-то там эвристического, который водит автомобиль по городу, например, а абсолютного, решающего все задачи на формальном языке.

Как вы думаете, существует ли такая программа:

10. На вход программы подается замкнутое синтаксически правильное предложение на формальном языке евклидовой геометрии (подробнее об этом языке — в приложении). Программа должна вернуть истинностное значение этого предложения.

Т.е. арифметика, как мы докажем, неразрешима. А разрешима ли геометрия? Оказывается, да (см. приложение). Оказывается, решать задачи по планиметрии можно, не задумываясь, по алгоритму.

11. Существует ли алгоритм, возвращающий «1», если в десятичной записи числа π есть последовательность из 20 семерок, и возвращающий «0», если такой последовательности нет.

Неправильное не-решение: Попытаемся построить такой алгоритм. Будем идти вдоль последовательности десятичных знаков π. Если нам встречается строка из 20 семерок подряд, то возвращаем «1». Но если не встречается, то не понятно, когда останавливаться...

Правильное решение: очевидно, существует. В десятичной записи π либо есть 20 семерок подряд, либо нет. Поэтому один из следующих двух алгоритмов гарантированно удовлетворяет условиям задачи `return 0;` или `return 1;`.

12. Существует ли алгоритм, решающий любую задачу из книги «Галочкин, Нестеренко, Шидловский. Введение в теорию чисел»?

Решение: очевидно, существует, т.к. задач в этой книге лишь конечное число.

2.2 Уточнение понятия «алгоритм»

Слушателям, возможно, не дает покоя вопрос, несуществование чего именно мы утверждаем, говоря о алгоритмически неразрешимых классах задач. Как формализовать наше интуитивное представление о бездумной, механической работе? Понятие алгоритма можно определить совершенно строго. Для этого нужно взять какой-нибудь конкретный, просто описываемый язык программирования, или конкретную формальную вычислительную систему (например, машину Тьюринга), и сказать, что алгоритм — это по определению этот язык или эта система. Например, можно сказать, что алгоритм — это по определению программа для машины Тьюринга.

В нашем курсе, однако, удобнее считать, что алгоритм — это по определению синтаксически правильный текст на некоторой абстрактной версии языка Си. Строго определять синтаксические правила этого языка и правила выполнения программ на нем можно, но долго. Поэтому выделим только сходства и различия между *C* и *Abstract C*.

В *Abstract C* можно использовать условные операторы, циклы, рекурсию, подпрограммы, переменные. Правда, только переменные типа int. Для простоты также запретим пользоваться указателями. Важное свойство синтаксиса языка *Abstract C* состоит в том, что должна существовать программа, определяющая, является ли входной текст (синтаксически правильной) программой на *Abstract C*.

Программы на языке *Abstract C* выполняются на некоторой абстрактной машине, которая может работать сколь угодно быстро, и у которой сколь угодно большая память (можно считать, что эта машина может увеличивать размер своей памяти в процессе выполнения программы или что память бесконечна, но не-нули записаны лишь в конечном числе ячеек). Обратите внимание: не бесконечно быстро, а сколь угодно быстро. Программы, впадающие в бесконечный цикл, не завершат свою работу никогда даже на сколь угодно быстрой машине.

Интересно то, что если вместо *Abstract C* взять какой-нибудь другой не слишком бедный формальный язык программирования, скажем Abstract LISP, то полученное понятие вычислимости будет в точности совпадать с первым! Тезис, утверждающий, что любая вычислимая в интуитивном смысле функция вычислима также машиной Тьюринга, называется тезисом Черча-Тьюринга. Как правило, в литературе для формализации понятия алгоритма используются более простые языки и системы: машина Тьюринга или машина Поста, нормальные алгоритмы Маркова, рекурсивные функции, лямбда-исчисление Черча.

2.3 Проблема остановки

В предыдущем разделе мы сформулировали проблему остановки для программ без входов. Алгоритмическую неразрешимость этой проблемы технически проще будет доказать для программ с ровно одним входом, а потом свести это к случаю программ без входов.

Теорема 1. *Пусть функция* $h\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *определена равенствами:* $h(n,m) = 0$, *если* $T_n(m) \neq *$, *и* $h(n,m) = 1$, *если* $T_n(m) = *$. *Тогда* h *невычислима.*

Доказательство. Будем доказывать от противного: предположим, что h вычислима. Тогда она реализуется некоторым алгоритмом H:

$$H(n,m) = \begin{cases} 0, \text{ если } T_n(m) \neq * \\ 1, \text{ если } T_n(m) = * \end{cases}$$

Построим теперь по H новый алгоритм H':

```
int main ( int n, int m ) {
   if ( H(n,m) == 0 ) return T_n(m);
   else return 1;
}
```

Заметим, что H' всегда останавливается. Значит, всегда останавливаться будет и алгоритм

$$H''(n) = H'(n,n) + 1$$

Но ведь H'' — это программа с одним входом! Значит,

$$H'' = T_k$$

для некоторого k, т.е.

$$H'(n,n) + 1 = T_k(n)$$

Вместо n подставим в это равенство k:

$$H'(k,k) + 1 = T_k(k)$$

Возможны два случая: либо $T_k(k)$ останавливается, либо нет. Если $T_k(k)$ останавливается, то получаем $T_k(k) + 1 = T_k(k)$, чего не может быть. Если $T_k(k)$ не останавливается, получаем не менее абсурдное равенство $1 + 1 = *$. Следовательно, исходное предположение о вычислимости h было неверным. ■

Получается, что написать такую программу-анализатор, определяющую, остановится ли входная программа, не возможно в принципе! Не помогут ни экзотические языки, ни метапрограммирование, ни ленивые вычисления, ни развитие науки. Доказательство алгоритмической неразрешимости проблемы остановки было придумано

Тьюрингом уже после основной работы Геделя. Скоро мы увидим, что оно очень похоже на одно из доказательств первой теоремы Геделя о неполноте.

13. Докажите, что проблема распознавания остановки для программ без входов тоже является алгоритмически неразрешимой.

14. Докажите, что существует такая программа A с одним входом, для которой проблема распознавания ее остановки на различных входах является алгоритмически неразрешимой.

15. Существует ли программа без входа, для которой проблема распознавания ее остановки алгоритмически неразрешима?

Решение: очевидно, нет, т.к. проблема остановки алгоритмически разрешима для любого конечного множества программ без входа. Аналогичная некорректная задача: может ли гипотеза Гольдбаха быть алгоритмически неразрешимой.

16. Назовем программу T_n с одним входом самоприменимой, если $T_n(n)$ останавливается. Докажите, что проблема распознавания самоприменимости является алгоритмически неразрешимой.

2.4 Как превзойти алгоритм

Хорошо, проблема остановки алгоритмически неразрешима. Но, может быть, нам и не нужен такой анализатор, который определяет, остановится или не остановится *любая* программа. Часто достаточно иметь алгоритм, который решает проблему остановки в 99% случаев (что бы это ни значило). Оказывается, для такого частичного распознавателя H мы всегда можем указать программу, относительно которой H не сможет сказать, остановится она или нет, но нам ответ будет известен!

Пусть H — такой частичный распознаватель:

$$H(n,m) = \begin{cases} 0, \text{ если } T_n(m) \neq * \\ 1 \text{ или } *, \text{ если } T_n(m) = * \end{cases}$$

Тогда, проводя для H предыдущее доказательство, мы по-прежнему получаем противоречие в случае $T_k(k) \neq *$, но случай $T_k(k) = *$ ничему не противоречит. Получается, что мы нашли программу T_k, на которой H не выдает ни 0, ни 1, но нам известно, что T_k не останавливается! Мы можем улучшить частичный распознаватель H: прописать в нем отдельной строчкой, что если $m = n = k$, то возвращать нужно 1. Получим другой частичный распознаватель H_1. Но к H_1 можно применить то же самое рассуждение и найти программу T_{k_1}, про которую нам тоже известно, что она не останавливается на входе k_1, но H_1 про $T_{k_1}(k_1)$ сказать ничего не сможет. Можно улучшить H_1 до H_2 и т.д..

Неформально этот результат можно интерпретировать так: представьте себе робота, решающего некоторые задачи. Тогда, изучив внутреннее устройство этого робота, можно подсунуть ему такую задачу, которую он не решит, но вам ответ на эту задачу будет известен! И как ни улучшай робота, все равно останутся задачи, относительно

которых правильный ответ нам будет известен, но роботу эти задачи будут не по зубам. Правда, это рассуждение годится только для достаточно широких классов задач типа проблемы остановки.

Интересна еще одна деталь: по H число k находится вычислимым образом! Можно представить себе двух роботов, H и G, которые будут генерировать друг для друга числа $k_1, l_1, k_2, l_2, \ldots$ и постоянно улучшать друг друга. Но если пару этих роботов рассматривать как один алгоритм F, то для F по-прежнему сгенерировать задачу m, для которой система из двух роботов не сможет дать ответ, хотя ответ будет известен внешнему наблюдателю.

2.5 Другие алгоритмически неразрешимые проблемы

В настоящее время известно немало алгоритмически неразрешимых классов задач. Вот некоторые примеры:

1. Проблема изоморфизма. Две конечнопорожденные группы G_1 и G_2 заданы своими образующими и определяющими соотношениями. Не существует алгоритма, определяющего, изоморфны ли G_1 и G_2. Более того, даже если задана одна группа G, не существует алгоритма, определяющего, изоморфна ли G тривиальной группе.

2. Неразрешимость предыдущего класс задач имеет интересное приложение в топологии. Известно, что для любой конечнопорожденной группы G найдется такое четырехмерное многообразие M^4, для которого $\pi_1(M^4) = G$. Генераторы и образующие $\pi_1(M^4)$ могут быть найдены вычислимым образом по триангуляции многообразия M^4. Следовательно, не существует алгоритма, определяющего, гомеоморфны ли два заданных своими триангуляциями четырехмерных многообразия M_1^4 и M_2^4. Для многообразий размерности 2 такой алгоритм существует. Не известно пока, существует ли он в размерности 3.

3. Проблема эквивалентности слов в ассоциативных исчислениях. Ассоциативное исчисление — это набор правил, по которым одни слова можно преобразовывать в другие, заменяя в них какие-то фиксированные подслова. Например, рассмотрим ассоциативное исчисление с алфавитом $\{a, b, c\}$ и правилами $bc = cb$, $ab = c$, $aa = \lambda$, где λ обозначает пустое слово. В этом исчислении слова $abab$ и bb будут эквивалентны, т.к. $abab = abc = acb = aabb = bb$. Для этого исчисления существует программа, по двум данным словам определяющая, эквивалентны они или нет. Однако известны такие ассоциативные исчисления, в которых проблема распознавания эквивалентности слов является алгоритмически неразрешимой.

4. Не существует алгоритма, верно отвечающего на вопрос о том, можно ли данным набором многоугольников замостить плоскость (само собой, эти многоугольники должны быть заданы какой-то строчкой символов конечного алфавита — этого

можно добиться, если ограничиться только многоугольниками с целочисленными координатами вершин).

5. В 1970 году, усилиями Девиса, Патнэма, Робинсон и Матиясевича, десятая проблема Гильберта была окончательно решена. Оказалось, что алгоритма, по данному диофантовому уравнению определяющего, имеет ли оно целочисленные решения, не существует!

Каким образом можно доказывать алгоритмическую неразрешимость подобных классов задач? В предыдущем разделе мы доказали алгоритмическую неразрешимость очень широкого класса задач: практически любую математическую задачу о текстах или целых числах можно переформулировать в виде «Остановится ли следующий алгоритм?». А классы задач из приведенного списка кажутся гораздо более узкими и специальными. Ключ к доказательству их алгоритмической неразрешимости — в возможности кодировать более-менее произвольные машины Тьюринга объектами задач из рассматриваемого класса.

Например, в (4) существует способ построения по каждой машине Тьюринга A набора многоугольников $\mathcal{P}(A)$, такой что A не завершает работу тогда и только тогда, когда многоугольниками из $\mathcal{P}(A)$ можно замостить всю плоскость. Интересно, что из доказательства алгоритмической неразрешимости (4) следует существование наборов многоугольников, используя которые замостить плоскость можно только непериодическим образом. Класс (3) получилось свести к алгоритмически неразрешимой проблеме распознавания доказуемости для некоторого формального языка: слова можно считать аксиомами, а подстановки — правилами вывода.

Слушатели, наверное, уже привыкли, что далеко не любая функция $f\colon \mathbb{N} \to \mathbb{N}$ является вычислимой и что алгоритмическая неразрешимость — не такая уж редкость. В математике бывают и другие виды неразрешимости: например, невозможность трисекции угла с помощью циркуля и линейки или неразрешимость уравнений пятой степени в радикалах.

17. Существует ли алгоритм распознавания эквивалентности слов в следующем ассоциативном исчислении? Алфавит: $\{a, b, c, d, e\}$. Допустимые подстановки: $ab = ba$, $ac = ca$, $ad = da$, $ae = ea$, $bc = cb$, $bd = db$, $be = eb$, $cd = dc$, $ce = ec$, $de = ec$.

18. Существует ли алгоритм распознавания эквивалентности слов в следующем ассоциативном исчислении? Алфавит: $\{a, b, c\}$. Допустимые подстановки: $b = acc$, $ca = accc$, $aa = \lambda$, $bb = \lambda$, $cccc = \lambda$. Подсказка: это ассоциативное исчисление группы симметрий квадрата, где a и b — симметрии относительно вертикальной и горизонтальной осей, а c — поворот на $90°$ против часовой стрелки.

19. Ассоциативное исчисление Цейтина задается алфавитом $\{a, b, c, d, e\}$ и правилами $ac = ca$, $ad = da$, $bc = cb$, $bd = db$, $eca = ce$, $edb = de$, $cca = ccae$. Эквивалентны ли в нем слова $abbcadebd$ и $deeabdceadcee$? Подсказка: все допустимые подстановки этого исчисления не меняют количество букв d в слове.

Что-то похожее на ассоциативное исчисление встретилось нам в курсе топологии, а именно в теореме о классификации двумерных многообразий. В том исчислении задача распознавания эквивалентности оказалась алгоритмически разрешимой. Скажем только, что алгебраическая топология дает гораздо более простые и быстрые алгоритмы определения рода (числа ручек или числа пленок Мебиуса) поверхности по ее триангуляции.

20. Замощение плоскости многоугольниками называется периодическим, если существует такое нетривиальное движение плоскости A, что для любого $n \in \mathbb{Z}$ преобразование A^n сохраняет это замощение. Приведите пример набора многоугольников, при помощи которого можно замостить плоскость

1) только периодическим образом;

2) и периодическим, и непериодическим образом.

2.6 Разрешимые и перечислимые множества

Определение 3. *Множество $A \subset \mathbb{N}$ называется разрешимым, если функция*

$$\chi_A(n) = \begin{cases} 0, \text{ если } n \notin A \\ 1, \text{ если } n \in A \end{cases}$$

вычислима.

Другими словами, множество $A \subset \mathbb{N}$ называется разрешимым, если существует алгоритм, по произвольному $n \in \mathbb{N}$ определяющий, принадлежит n множеству A или не принадлежит.

Утверждение 1. *Свойства разрешимых множеств:*

1) Если $A \subset \mathbb{N}$ и $B \subset \mathbb{N}$ разрешимы, то $A \cap B$ разрешимо.

2) Если $A \subset \mathbb{N}$ и $B \subset \mathbb{N}$ разрешимы, то $A \cup B$ разрешимо.

3) Если $A \subset \mathbb{N}$ разрешимо, то $\mathbb{N} \backslash A$ разрешимо.

Доказательство. Тривиально. Например, характеристическая функция множества $A \cup B$ реализуется следующим алгоритмом:

```
bool main ( int n ) {
   if ( A(n) || B(n) ) return 1;
   else return 0;
}
```

■

Т.е. разрешимые множества образуют коммутативное и ассоциативное кольцо относительно сложения $A \Delta B$, умножения $A \cap B$ и с единицей $\mathbb{N}$.

Определение 4. *Множество $A \subset \mathbb{N}$ называется перечислимым, если A является множеством значений вычислимой последовательности натуральных чисел.*

Другими словами, множество $A \subset \mathbb{N}$ называется перечислимым, если существует алгоритм, генерирующий один за другим (не обязательно в порядке возрастания) все элементы множества A. Например, множество всех четных натуральных чисел перечислимо, т.к. его генерирует следующий алгоритм:

```
void main () {
   int n = 1;
   while ( true ) {
      Write ( 2*n );
      n++;
   }
}
```

21. Докажите, что множество $A \subset \mathbb{N}$ является перечислимым тогда и только тогда, когда его полухарактеристическая функция

$$\chi_A^*(n) = \begin{cases} *, \text{ если } n \notin A \\ 1, \text{ если } n \in A \end{cases}$$

вычислима.

Утверждение 2. *Каждое разрешимое множество перечислимо.*

Доказательство. Тривиально. Пусть A — произвольное разрешимое множество и пусть его характеристическая функция χ_A реализуется программой `A ( int n )`. Напишем тогда программу, перечисляющую A:

```
void main () {
   int n = 1;
   while ( true ) {
      if ( A(n) ) Write (n);
      n++;
   }
}
```

■

Утверждение 3. *Свойства перечислимых множеств:*
1) Если $A \subset \mathbb{N}$ и $B \subset \mathbb{N}$ перечислимы, то $A \cap B$ перечислимо.
2) Если $A \subset \mathbb{N}$ и $B \subset \mathbb{N}$ перечислимы, то $A \cup B$ перечислимо.
3) Если множества A и $\mathbb{N} \backslash A$ перечислимы, то A разрешимо.

Доказательство. Тривиально. Докажем только часть (3). Есть две программы, перечисляющие множества A и $\mathbb{N} \backslash A$. Как по ним построить программу, вычисляющую характеристическую функцию множества A? Запустим эти две программы параллельно. Они будут генерировать две последовательности натуральных чисел. Любое

$n \in \mathbb{N}$ рано или поздно встретится либо в первой последовательности, либо во второй. Если n встретилось в первой, возвращаем «1», если во второй — «0».■

По тем же мощностным соображениям, по которым существуют невычислимые числа и невычислимые функции, существуют и неразрешимые, и неперечислимые множества. Действительно, множество всех подмножеств множества натуральных чисел не счетно. Бывают ли перечислимые, но не разрешимые множества? Мощностные соображения здесь уже не помогут. К счастью, у нас уже все готово для того, чтобы привести конкретный пример. Заодно мы приведем пример конкретного неперечислимого множества.

Как обычно, занумеруем множество синтаксически правильных программ без входа каким-нибудь вычислимым образом. Пусть S — множество номеров останавливающихся программ, или просто множество останавливающихся программ. Тогда $\mathbb{N}\backslash S$ — множество неостанавливающихся программ.

Утверждение 4. *Множество S перечислимо, но не разрешимо. Множество $\mathbb{N}\backslash S$ неперечислимо.*

Доказательство. Неразрешимость S — это в точности утверждение о алгоритмической неразрешимости проблемы остановки, которое мы уже доказали. Как перечислить S? Будем идти вдоль натурального ряда и запускать программы с номерами $1, 2, 3 \ldots$, но довольно хитрым образом: мы не ждем, пока программа с номером 2 закончит работу (она может ее и не закончить), а потом запускаем программу с номером 3. Мы делаем так: когда доходим до 2, запускаем программу с номером 2, но только на один шаг ее работы. Потом переходим к 3, запускаем 3 на один шаг и сразу после этого делаем второй шаг работы программы 2. Потом переходим к 4 и т.д.. Т.е. получается, что одновременно работают сразу несколько программ. Если вдруг какая-то программа с номером n заканчивает работу, то пусть исходная программа выводит номер n: `Write (n);`. Очевидно, все множество S будет таким образом перечислено.

Множество $\mathbb{N}\backslash S$ неперечислимо, потому что в противном случае, в силу утверждения 3, множество S было бы разрешимо. ■

3 Логика предикатов

3.1 О том, что есть

Во введении мы убедились, что последовательностям букв не всегда получается придать смысл и истинностное значение. Но, если мы хотим создать такой формальный язык, в котором любое синтаксически правильное предложение имеет смысл, то нам необходимо выяснить, какие именно особенности языка могут этой цели помешать. Во введении приведен пример одной такой особенности: возможность предложениям языка ссылаться на самих себя. Но есть и другие препятствия.

Ниже приведен фрагмент одного из доказательств существования Бога из трактата «Прослогион» (1077 год) средневекового философа Ансельма Кентерберийского:

> Итак, Господи, дающий вере разумение, дай мне, сколько сам знаешь, чтобы я понял, что Ты есть, как мы веруем, и то есть, во что мы веруем. А мы веруем, что Ты есть нечто, больше чего нельзя ничего себе представить. Значит, когда сказал безумец в сердце своем: «нет Бога» — он сказал, что какой-то такой природы нет? Но, конечно, этот же самый безумец, слыша, как я говорю: «Нечто, больше чего нельзя ничего себе представить», — понимает то, что слышит; а то, что он понимает, есть в его уме, даже если он не имеет в виду, что такая вещь существует. [...] И, конечно, то, больше чего нельзя себе представить, не может быть только в уме. Ибо если оно уже есть по крайней мере только в уме, можно представить себе, что оно есть и в действительности, что больше. Значит, если то, больше чего нельзя ничего себе представить, существует только в уме, тогда то, больше чего нельзя себе представить, есть то, больше чего можно представить себе. Но этого, конечно, не может быть. Итак, без сомнения, нечто, больше чего нельзя себе представить, существует и в уме, и в действительности.

Это рассуждение несколько запутано, но его суть можно передать кратко: «Бог — это по определению существующее, всемогущее, всеблагое и т.д.. Следовательно, он существует». Вряд ли кто-то признает такое рассуждение правильным доказательством. Но где именно ошибка? Каков должен быть язык, чтобы все синтаксически правильные рассуждения на нем автоматически были корректными доказательствами? Бог тут ни при чем — та же проблема может возникнуть в рассуждениях о гораздо более приземленных вещах: о пегасах, например. Мы можем определить пегаса как существующую крылатую лошадь. Следовательно, пегасы существуют. Доказательства, подобные этому доказательству Ансельма в том или ином виде повторялись еще в течение многих веков.

Первое, что приходит в голову — запретить использование предиката (прилагательного) «существующий» и оставить только специальный глагол «существует», который еще нужно употреблять очень аккуратно — не как "обычные"глаголы типа «летит», «видит» и т.д.. По этому пути пошел Кант, который отнес существование к одной группе категорий, а предикаты — к другой (впрочем, еще Аристотель указывал на то, что сущее не является максимальным родом). Однако проблемы существования на этом не закончились.

Пегасы должны в каком-то смысле быть, иначе чего же, собственно, нет? Может быть, они существуют лишь в воображении, а не в физическом мире? А как тогда быть, например, с «круглым квадратом»? Его даже в воображении нет. Опять же, чего именно нет? Чтобы сказать, что круглого квадрата не существует, мы должны считать, что в каком-то смысле он все же есть, иначе чего же именно не существует?

Попытку решения этой проблемы мы находим уже в диалогах Платона. Современный философ Куайн назвал ее проблемой «бороды Платона» — из-за того, что она очень эффективно затупляет бритву Оккама. В начале двадцатого века решение проблемы бороды Платона было найдено. Оно потребовало радикальных изменений в языке: нужно было полностью отказаться от существительных (и от субъектно-

предикатной аристотелевской логики)!

Давайте вместо того, чтобы задумываться, в каком смысле существуют пегасы, выясним, истинно или ложно следующее предложение:

$$(1) \qquad \exists x(x \text{ — крылатая лошадь})$$

Это предложение, конечно, ложно, если под существованием подразумевать существование в физическом мире. Но это мы знали и раньше. Изменение же в том, что, если (1) ложно, то не нужно задумываться, в каком смысле существуют пегасы. Потому что мы нигде не используем существительное «пегас». «Крылатая лошадь» же здесь является не существительным, а предикатом: являться крылатой лошадью. Относительно любого предмета физического мира x мы способны определить, является ли x крылатой лошадью. Решение проблемы бороды Платона достигается из-за того, что, если мы считаем предложение (1) истинным, то мы обязываем себя принять онтологию, в которой Пегасы существуют, но мы не обязываем себя ее принимать, когда говорим, что (1) ложно.

Как следствие, в предложениях на формальном языке, который мы будем строить, отдельные подстроки легко могут быть бессмысленны, тогда как все предложение целиком будет иметь смысл. В предложении на русском языке

$$(4) \qquad \text{«белая кошка сидит на подоконнике»?}$$

отдельные подстроки «кошка», «белая кошка», «подоконник» вроде бы имеют смысл: если мы видим набор букв «кошка», то представляем себе кошку. Люди пытались даже придать смысл подстрокам «белая», «сидит» и «на», сводя их так или иначе к существительным, например, к «белизне». В формальном языке арифметики существительных не будет, и поэтому подстрокам

$$(2|n) \wedge (n \neq 2) \qquad \text{или} \qquad \exists l(k \text{ — простое}) \wedge (l \text{ — простое}) \wedge (n = k + l)$$

осмысленного предложения

$$(A) \qquad \text{«}\forall n\big((2|n) \wedge (n \neq 2) \to \exists k \exists l(k \text{ — простое}) \wedge (l \text{ — простое}) \wedge (n = k + l)\big)\text{»}$$

не получается придать какой-либо смысл, это всего лишь последовательности символов.

22. Переведите на «язык без существительных» следующие предложения:

1) Все судьи — юристы.
2) Среди адвокатов нет судей.
3) Юристы — это не только судьи и адвокаты.
4) Максим Горький и Алексей Пешков — это один и тот же человек.

3.2 Формальные языки первого порядка

В этом разделе мы определим формальный язык с формальным синтаксисом, на котором можно будет записывать математические утверждения. Какие языковые конструкции для этого понадобятся? Посмотрим на неформальную пока запись типичного математического утверждения:

$$\forall n\big((2|n) \wedge (n \neq 2) \to \exists k \exists l (k \text{ — простое}) \wedge (l \text{ — простое}) \wedge (n = k + l)\big)$$

Хочется, чтобы наш язык "поддерживал":

1. Переменные: n, k, l.
2. Кванторы: $\forall n$, $\exists k$.
3. Логические связки: $\wedge$, $\to$.
4. Предикаты: «являться простым числом», $=$.
5. Функции: $+$. Под предикатами будем иметь в виду отображения из некоторого множества M во множество $\{0, 1\}$ истинностных значений. Под функциями — отображения из M или декартовой степени M в само же множество M. Например, «являться простым числом» — это отображение $\mathbb{N} \to \{0, 1\}$, равенство — это отображение $\mathbb{N} \times \mathbb{N} \to \{0, 1\}$, а сложение — это отображение $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Символы $+$ и $=$ только для удобства поставлены между переменным, а не перед ними. С тем же успехом можно было использовать запись $=(2, 3) = 0$, $+(3, 8) = 11$. Обратите внимание, что знак равенства в формуле $+(3, 8) = 11$ и второй знак равенства в формуле $=(2, 3) = 0$ — это равенства уже в метаязыке.

Прежде, чем определять формальный язык, необходимо определить алфавит. Пусть наш алфавит состоит из маленьких букв латинского алфавита (переменные), скобок, символов $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$, $\forall$, $\exists$. Правда, в латинском алфавите всего лишь конечное число букв. И поэтому, чтобы в предложениях можно было использовать сколь угодно большое количество переменных или предикатных символов, добавим в алфавит еще символ «$'$». Тогда переменные можно будет обозначать так: x, x', x'' и т.д..

Следующий шаг: определим, какие функциональные и предикатные символы будут использоваться в языке. Т.е. добавим в алфавит еще несколько символов (например, «является простым числом» и «+»), но просто добавим их, а еще запомним их валентность — сколько переменных должно идти после них, чтобы предложение было синтаксически правильным. Например, разумно положить валентность предикатного символа «является простым числом» равной одному, а валентность функционального символа $+$ — равной двум. Произвольный набор предикатных и функциональных символов с указанием их валентности называется *сигнатурой* языка.

Теперь нужно определить, какие конечные последовательности букв (выражения) являются синтаксически правильными. Это определение будет рекурсивно. Для начала определим, что такое терм:

1. Переменная есть терм.
2. Если $t_1, \ldots, t_k$ — термы, а f — функциональный символ валентности $k \geqslant 0$, то выражение $f(t_1, \ldots, t_k)$ есть терм.

Если P — предикатный символ валентности $k \geqslant 0$, а $t_1, \ldots, t_k$ — термы, то выражение $P(t_1, \ldots, t_k)$ называется атомарным предложением. Наконец, определим, что такое (синтаксически правильное) предложение:

1. Атомарное предложение есть предложение.
2. Если φ — предложение, то $\neg\varphi$ — предложение.
3. Если φ и ψ — предложения, то выражения $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, $(\varphi \leftrightarrow \psi)$ также являются предложениями.
4. Если φ — предложение, а x — переменная, то выражения $\forall x\varphi$ и $\exists x\varphi$ есть предложения.

Нам понадобится одно очевидное утверждение:

Утверждение 5. *Задача определения по произвольной конечной последовательности букв алфавита, является ли эта последовательность синтаксически правильным предложением, алгоритмически разрешима.*

Поэтому в дальнейшем будем рассматривать только синтаксически правильные предложения и называть их просто предложениями.

Пример 1. Сигнатура частично упорядоченного множества состоит из двух двуместных предикатных символов: $=$ и $\leqslant$. В этой сигнатуре можно записать, например, такое предложение:

$$\forall x \forall y (((x \leqslant y) \wedge (y \leqslant x)) \to (x = y))$$

Или такое:

$$\forall x \exists z (x \leqslant y)$$

Оба будут синтаксически правильными. А вот пример синтаксически неправильного предложения:

$$)\forall = zz \leqslant))y \vee x\exists$$

Пример 2. Сигнатура группы состоит из двуместного функционального символа $\cdot$, одноместного функционального символа inv, нульместного функционального символа e (можно считать, что переменные запрещается обозначать буквой e) и двуместного предикатного символа $=$. Пример синтаксически правильного предложения этой сигнатуры:

$$\forall x (\mathrm{inv}(\mathrm{inv}(x)) = x)$$

Пример синтаксически неправильного:

$$\forall x \forall y \exists z (z = \mathrm{inv}(x, y))$$

Пример 3. Сигнатура арифметики состоит из двух двуместных функциональных символов $+$ и $\cdot$ и одного двуместного предикатного символа $=$.

Пример 4. Сигнатура теории множеств состоит из двух двуместных предикатных символов: $=$ и $\in$. А значит, ничем не отличается от сигнатуры первого примера. Различие между теорией множеств и частично упорядоченным множеством состоит не в том, какие предложения являются синтаксически правильными, а в том, какой смысл мы придаем символам $\leqslant$ и $\in$. Другими словами, различие между ними не в синтаксисе, а в семантике.

3.3 Семантика языков первого порядка

Чтобы можно было говорить об истинности предложений, нужно каким-то образом интерпретировать входящие в сигнатуру пропозициональные и функциональные символы, а также определить, элементам какого множества соответствуют переменные. Пусть Ω — произвольная сигнатура языка первого порядка. Для того, чтобы задать *интерпретацию* $\mathcal{M}$ сигнатуры Ω, необходимо:

1. Указать некоторое непустое множество M, называемое носителем интерпретации.

2. Каждому предикатному символу валентности k поставить в соответствие k-местный предикат, т.е. отображение $M^k \to \{0,1\}$.

3. Каждому функциональному символу валентности k поставить в соответствие функцию k переменных, т.е. отображение $M^k \to M$ (если $k=0$, то $M^k = \{\emptyset\}$, т.е. функции от нуля переменных — это элементы M, константы).

Пример 5. Рассмотрим три различных интерпретации сигнатуры $(\leqslant, =)$ теории частично упорядоченного множества.

1. Пусть носителем интерпретации $\mathcal{M}_1$ будет множество $M_1 = \mathbb{Z}$, предикатный символ $=$ обозначает совпадение элементов в $\mathbb{Z}$, а символу $\leqslant$ поставлен в соответствие двуместный предикат $P\colon \mathbb{Z}\times\mathbb{Z} \to \{0,1\}$, такой что $P_{\leqslant}(m,n) = 1$ тогда и только тогда, когда $m \leqslant n$.

2. $\mathcal{M}_2$ определим точно так же, но с $\mathbb{Q}$ вместо $\mathbb{Z}$.

3. Пусть носителем $\mathcal{M}_3$ будет множество $M_3 = \mathbb{R}$; $P_{=}(x,y) = 1$ тогда и только тогда, когда $|x-y| = 10$; $P_{\leqslant}(x,y) = 1$ тогда и только тогда, когда $x^2 + y^2 = 1$.

Если фиксирована интерпретация $\mathcal{M}$ сигнатуры Ω, то уже имеет смысл говорить об истинности предложений. Рассмотрим, например, такое предложение в сигнатуре частично упорядоченного множества:

$$(*) \qquad \forall x \forall y (((x \leqslant y) \wedge \neg(x = y)) \to \exists z ((x \leqslant z) \wedge (z \leqslant y) \wedge \neg(z = x) \wedge \neg(z = y)))$$

Оно истинно в интерпретации $\mathcal{M}_2$, но ложно в $\mathcal{M}_1$ и $\mathcal{M}_3$. Если предложение φ истинно в интерпретации $\mathcal{M}$, будем обозначать это так: $\mathcal{M} \models \varphi$.

Мы, правда, забыли одну деталь. Не всякому синтаксически правильному предложению, даже если задана интерпретация, сопоставлено какое-то истинностное значение. Например, бессмысленно говорить об истинности предложения

$$x + 3 = 15,$$

даже если мы фиксировали интерпретацию (скажем, $M = \mathbb{N}$, $+$ означает сложение натуральных чисел, $=$ означает совпадение двух натуральных чисел). А вот об истинности предложений

$$\forall x(x + 3 = 15) \qquad \text{или} \qquad \exists x(x + 3 = 15)$$

говорить уже можно. Предложение $x + 3 = 15$ плохо тем, что в нем есть одна переменная x, про которую не понятно, что утверждается: что для любого x или что существует x? Вхождение переменной (т.е. конкретный символ переменный), которое не лежит в области действия ни одного квантора по этой переменной, называется *свободным*. Назовем предложение *замкнутым*, если в нем нет свободных вхождений переменных.

23. Истинно ли предложение $x = x$ в интерпретации $(\mathbb{Z}, =)$?

Решение: нет, истинность для этого предложения не определена, т.к. оно не замкнуто: переменная x не входит в область действия никакого квантора.

Итак, одно и то же замкнутое предложение может быть истинно в одной интерпретации и ложно в другой. Оказывается, бывают предложения, истинные в любой интерпретации. Пример:

$$\forall x \forall y(P(x) \to (P(x) \vee P(y)))$$

Это предложение будет истинным вне зависимости от того, что такое P и какое множество пробегают x и y. Предложения, истинные в любой интерпретации, называются *общезначимыми* или *тавтологиями*. Общезначимость предложения φ будем обозначать так: $\models \varphi$. Предложения, ложные в любой интерпретации, называются *противоречивыми*.

24. Пусть P — двуместный предикатный символ. Какие из следующих предложений являются общезначимыми?

1) $\forall x \exists y P(x, y) \to \exists y \forall x P(x, y)$;
2) $\exists x \forall y P(x, y) \to \forall y \exists x P(x, y)$.

25. Для каждой пары этих интерпретаций приведите пример предложения соответствующей сигнатуры, истинного в одной интерпретации и ложного в другой:

1) $\mathbb{N}$ и $\mathbb{Z}$, $\Omega = \{<\}$
2) $\mathbb{Q}$ и $\mathbb{Z}$, $\Omega = \{<\}$

3) $\mathbb{N}$ и $\mathbb{Z}$, $\Omega = \{+\}$
4) $\mathbb{Q}$ и $\mathbb{Z}$, $\Omega = \{+\}$
5) $\mathbb{Q}$ и $\mathbb{R}$, $\Omega = \{+, \cdot\}$

26. Приведите пример предложения в сигнатуре группы

6) истинное в любой коммутативной группе и ложное в любой некоммутативной;

7) истинное в любой группе, изоморфной $\mathbb{Z}_5$, и ложное в любой другой группе. Существует ли предложение, истинное в любой конечной группе и ложное в любой бесконечной?

Очень часто в рассматриваемых нами сигнатурах будет встречаться двуместный предикатный символ $=$, о котором не хотелось бы каждый раз вспоминать и отдельно описывать, как его интерпретировать. Поэтому далее мы будем рассматривать только т.н. *нормальные* модели, где равенство всегда интерпретируется как совпадение элементов в носителе модели.

Слушатели, конечно, заметили, что для возможности рассуждений о интерпретациях необходимо встать, хотя бы частично, на платонистическую позицию. Возьмем, например, сигнатуру языка теории множеств и интерпретируем ее так: M — это класс всех множеств, $\in$ — отношение принадлежности, $=$ — совпадение множеств. Но что такое множество? Выход — рассматривать только относительно «надежные» интерпретации. Мало кто усомнится, что любое замкнутое предложение в сигнатуре $(+, \cdot)$, интерпретируемое через натуральные числа, будет либо истинным, либо ложным.

27. Докажите, что следующее предложение ложно во всех интерпретациях с конечным носителем:

$$\forall x \forall y \forall z (P(x,y) \wedge (P(y,z) \to P(x,z))) \quad \wedge \quad \forall x \neg P(x,x) \quad \wedge \quad \forall x \exists y P(x,y)$$

3.4 Выразимость предикатов

В предыдущих разделах было сказано, что сигнатура формального языка арифметики состоит только из двух двуместных функциональных символов $+$ и $\cdot$ (и двуместного предикатного символа $=$, но о равенстве мы больше не будем говорить специально). Однако в записи гипотезы Гольдбаха встречаются и другие предикатные символы, например, «является простым числом» или «является четным числом». Конечно, можно было расширить сигнатуру, включив туда эти символы, а также другие: нульместный функциональный символ 1, предикатный символ «является точным квадратом» и т.д.. Но можно этого и не делать. Действительно, предложение «n делится на 2» эквивалентно такому:

$$\exists m (n = m + m)$$

Дадим точное определение. Пусть фиксирована сигнатура Ω и ее интерпретация $\mathcal{M}$. Пусть φ — синтаксически правильное предложение сигнатуры Ω с k свободными переменными $x_1, \ldots, x_k$. Истинность φ в интерпретации $\mathcal{M}$ зависит только от значений

переменных $x_1, \ldots, x_k$. Таким образом, определено отображение $P_\varphi \colon M^k \to \{0,1\}$. Предикат P называется *выразимым*, если $P = P_\varphi$ для некоторого φ. Множество $A \subset M$ называется выразимым, если существует такой одноместный выразимый предикат P, что $a \in A$ тогда и только тогда, когда $P(a) = 1$.

Другими словами, предикат выразим в языке, если его можно описать предложением этого языка. Например, предикат «является четным числом» выразим в $(\mathbb{N}, +, \cdot)$. Можно еще привести такую аналогию: представьте, что вы столкнулись с внеземной цивилизацией. Инопланетяне понимают, что такое $\mathbb{N}$, что такое $+$ и что такое $\cdot$. Сможете ли вы им объяснить, что такое четное число? Да, конечно: четные числа — это в точности натуральные числа, раскладывающиеся в сумму двух равных натуральных чисел. Сможете ли вы объяснить, что такое 1? Тоже да, т.к. 1 характеризуется тем свойством, что если любое число домножить на 1, то оно не изменится. На научном языке это можно сказать так: предикат $n = 1$ выразим в $(\mathbb{N}, +, \cdot)$ предложением

$$\forall m(mn = m)$$

Правда, эта аналогия актуальна только при одном допущении: если в объяснениях ограничиваться только логикой первого порядка. Что такое логика не-первого порядка? В школе операцию умножения натуральных чисел объясняют через сложение: чтобы перемножить числа m и n, нужно n раз сложить m с самим собой. Следуя этой идее, можно попробовать выразить трехместный предикат $mn = k$ в $(\mathbb{N}, +)$ предложением

$$m + \ldots + m(n \text{ раз}) = k$$

Но это предложение не является предложением на языке первого порядка! В нем делается ужасная вещь: переменная n участвует в описании самого предложения! И неизвестно, к каким парадоксам это может привести. Поэтому в настоящем тексте мы изучаем только языки первого порядка.

28. Выразите следующие предикаты в $(\mathbb{N}, +, \cdot)$:

1) n — простое;
2) n представимо в виде суммы трех квадратов;
3) $n \leqslant m$;
4) $n = 14$;
5) $n|m$;
6) a и b — взаимно простые;
7) q есть частное при делении a на b;
8) r есть остаток при делении a на b;
9) n — степень двойки. Подсказка: степени двойки характеризуются тем, что все их неединичные делители четны.

29. Перепишите гипотезу Гольдбаха в сигнатуре $(+, \cdot)$.

Существуют ли невыразимые в $(\mathbb{N}, +, \cdot)$ множества? Из мощностных соображений следует, что существуют (существует лишь счетное число предложений, а подмно-

жеств в $\mathbb{N}$ континуум). Но как привести пример? Ведь мы по сути не знаем о натуральных числах ничего, кроме сложения и умножения? Мы даже увидим, что язык $(+, \cdot)$ настолько богат, что выразимо любое перечислимое множество! Скоро будет доказано, что даже неперечислимое множество $\mathbb{N}\backslash S$ неостанавливающихся программ выразимо! Сможем ли мы придумать множество, еще более ужасное, чем $\mathbb{N}\backslash S$? Оказывается, с таким множеством мы имеем дело каждый день.

30. Выразите следующие предикаты в соответствующих интерпретациях:
1) $n = m + 1$ в $(\mathbb{N}, <)$;
2) $n = m + 2$ в $(\mathbb{Z}, <)$;
3) $xy = z$ в $(\mathbb{R}, +, y = x^2)$.

31. Рассмотрим сигнатуру с двуместным предикатным символом P и ее нормальную интерпретацию с носителем $\mathbb{R}^2$ и предикатом «находятся на расстоянии 1». Выразите предикаты «находятся на расстоянии 2», «находятся на расстоянии не более 2», «находятся на расстоянии 1/2».

32. Рассмотрим сигнатуру с трехместным предикатным символом C и ее нормальную интерпретацию с носителем $\mathbb{R}^2$, где $C(x, y, z)$ означает, что точки x и y равноудалены от точки z. Выразите следующие предикаты:

1) Точки A, B, C лежат на одной прямой. Подсказка: это означает, что не существует другой точки C', находящейся на тех же расстояниях от A и B, что и точка C.

2) Прямые AB и CD параллельны.

3) Точки A, B, C и D являются вершинами параллелограмма.

4) $|AB| = |CD|$.

5) $|OA| \leqslant |OB|$. Подсказка: Напишите, что все прямые, проходящие через A, пересекаются с окружностью радиуса OB с центром O.

Утверждение 6. *Свойства выразимых в $(\Omega, \mathcal{M})$ множеств:*
1) Если $A \subset M$ и $B \subset M$ выразимы, то $A \cap B$ выразимо.
2) Если $A \subset M$ и $B \subset M$ выразимы, то $A \cup B$ выразимо.
3) Если $A \subset M$ выразимо, то $M\backslash A$ выразимо.

Доказательство. Тривиально. Например, если предложения φ и ψ со свободными переменными y и z выражают множества A и B соответственно, то предложение $\varphi \wedge \psi$, в котором все свободные вхождения y и z заменены на x, выражает множество $A \cap B$. ■

Следующие две задачи относятся не к выразимости предикатов, а к некоторым другим уровням выразимости.

33. Инопланетяне знают только один квантор: существование (и знают все логические связки). Сможем ли мы им объяснить, что такое $\forall$? Сможем ли мы объяснить, что такое $\exists$, если они знают только $\forall$?

Решение: сможем, т.к. предложения $\forall x P(x)$ и $\neg\exists x \neg P(x)$ истинны в одних и тех же интерпретациях (эквивалентны). Предложения $\exists x P(x)$ и $\neg\forall x \neg P(x)$ тоже эквивалентны.

34. Инопланетяне знают, что такое «и» и «не». Сможем ли мы им объяснить, что такое «или»? Сможем ли мы объяснить любую логическую связку (булеву функцию двух переменных)? Тот же вопрос для наборов $\{\vee, \neg\}$, $\{\to, \neg\}$, $\{\to, 0\}$ и наборов $\{\vee, \wedge\}$, $\{\vee, \wedge, \leftrightarrow\}$, $\{\vee, \wedge, 0, 1\}$.

3.5 Автоморфизмы интерпретаций и невыразимость

Как можно доказывать невыразимость предикатов?

35. Выразим ли предикат $x = 0$ в $(\mathbb{Z}, \leqslant)$?

Кажется, что нет, потому что можно сдвинуть $\mathbb{Z}$ на единицу вправо: такой сдвиг не повлияет на отношение $\leqslant$, но переведет 0 в не-0. Дадим точные определения.

Пусть $\mathcal{M}_1 = (M_1, \{P_1^\alpha\}, \{f_1^\beta\})$ и $\mathcal{M}_2 = (M_2, \{P_2^\alpha\}, \{f_2^\beta\})$ — две интерпретации сигнатуры Ω. *Изоморфизмом* интерпретаций $\mathcal{M}_1$ и $\mathcal{M}_2$ называется отображение $h\colon M_1 \to M_2$, такое что

1. h — биекция.
2. Для любого α и любых $x_1, \ldots, x_k \in M_1$ выполняется

$$P_1^\alpha(x_1, \ldots, x_k) = P_2^\alpha(h(x_1), \ldots, h(x_k))$$

3. Для любого β и любых $x_1, \ldots, x_k \in M_1$ выполняется

$$h(f_1^\beta(x_1, \ldots, x_k)) = f_2^\beta(h(x_1), \ldots, h(x_k))$$

Последние два условия можно более кратко переписать на языке коммутативных диаграмм:

$$P_2^\alpha \circ h = P_1^\alpha, \qquad f_2^\beta \circ h = h \circ f_1^\beta$$

Автоморфизмами называют изоморфизмы $\mathcal{M} \to \mathcal{M}$. Предикат P интерпретации $\mathcal{M}$ называется устойчивым относительно автоморфизмов этой интерпретации, если для любого автоморфизма h выполнено $P \circ h = P$. Достаточно очевидно следующее утверждение (его строгое доказательство требует индукции по построению предложения, выражающего предикат):

Утверждение 7. *Предикат, выразимый в данной интерпретации, устойчив относительно ее автоморфизмов.*

36. Докажите, что следующие предикаты невыразимы в соответствующих интерпретациях:

1) $x = 1$ в $(\mathbb{Q}, <, +)$
2) $x = 1/2$ в $(\mathbb{R}, <, 0, 1)$

3) $x < y$ в $(\mathbb{Z}, +)$
4) $z = 0$ в $(\mathbb{Z}, y = x + 1)$
5) $x = 2$ в $(\mathbb{N}, \cdot)$
6) $x = \sqrt{2}$ в $(\mathbb{R}, +, 0, 1)$

Очень часто у интерпретаций нет нетривиальных автоморфизмов. Рассмотрим сигнатуру с равенством, одноместным функциональным символом S и ее интерпретацию с носителем $\mathbb{N}$, где S означает прибавление единицы.

37. Выразите в $(\mathbb{N}, S)$ следующие предикаты:
1) $y = x + 2$
2) $x = 0$
3) $x = 1$

Тем не менее, выразить в $(\mathbb{N}, S)$ стандартные множества, выразимые в $(\mathbb{N}, +, \cdot)$ (типа множества четных чисел или множества простых чисел), как правило, не получается. Однако метод доказательства невыразимости с помощью автоморфизмов интерпретации здесь не поможет!

38. Докажите, что у интерпретации $(\mathbb{N}, S)$ нет нетривиальных автоморфизмов.

39. Докажите, что любое предложение в $(\mathbb{N}, S)$ эквивалентно некоторому бескванторному предложению в $(\mathbb{N}, S, 0)$. Выведите отсюда, что в $(\mathbb{N}, S)$ выразимы только одноместные предикаты вида $x = a$ или двуместные предикаты вида $y = x + a$, где a — какое-то натуральное число.

40. Как в интерпретации $(\mathbb{N}, S)$ выразить предикат $x = 1000000$ предложением, которое поместится на лист бумаги?

41. Докажите, что у интерпретации $(\mathbb{N}, +)$ нет нетривиальных автоморфизмов. Приведите пример нетривиального автоморфизма $(\mathbb{N}, \cdot)$.

3.6 Выразимость в арифметике

В интерпретации $(\mathbb{N}, +, \cdot)$ мы уже выразили немало предикатов: $x = 0$, $x = 1$, $x = 100$, $x < y$, $x|y$, x — простое, x — степень двойки и т.д.. Оказывается, сложения и умножения достаточно достаточно для того, чтобы выразить любое перечислимое множество! Для доказательства этого утверждения нужно придумать какой-то метод кодирования произвольной машины Тьюринга некоторым арифметическим предикатом. Частная задача: научиться кодировать последовательности натуральных чисел произвольной длины, не прибегая к логике второго порядка.

Определим т.н. β-функцию Геделя. Пусть a, b, i — натуральные числа (для удобства 0 в настоящем тексте тоже считается натуральным числом) и $\beta(a, b, i)$ есть остаток при делении a на $(i + 1)b + 1$. Предикат $\beta(a, b, i) = k$, очевидно, выразим.

Лемма 1. *Для любого l и любой конечной последовательности $k_0, k_1, \ldots, k_{l-1}$, существует b, большее всех элементов этой последовательности и такое, что числа $b + 1, 2b + 1, \ldots, lb + 1$ попарно взаимно просты.*

Доказательство. Положим $b = l! \cdot (\max\{k_0, \ldots, k_{l-1}\} + 1)$. От противного: предположим, что для некоторых $i, j < l$ числа $ib+1$ и $jb+1$ не взаимно просты и делятся на некоторое простое число p. Тогда и их разность $b(i-j)$ делится на p. В любом случае b делится на p (если $i-j$ делится на p, то $p < l$ и, следовательно, b делится на p). Но тогда $ib+1$ не делится на p. Противоречие.■

Утверждение 8. *Для любого l и любой конечной последовательности $k_0, k_1, \ldots, k_{l-1}$, существуют такие a и b, что $\beta(a, b, i) = k_i$ для любого индекса $i = 0, \ldots, l-1$.*

Доказательство. Получается из предыдущей леммы применением китайской теоремы об остатках.■

Теперь мы умеем кодировать произвольную конечную последовательность натуральных чисел с помощью всего трех чисел a, b и $l!$ Как это поможет в выражении предикатов? Рассмотрим, например, трехместный предикат $x^y = z$. Может показаться, что его не выразить в $(\mathbb{N}, +, \cdot)$, ограничиваясь только средствами первого порядка, подобно тому как предикат $x \cdot y = z$ не выразим в $(\mathbb{N}, +)$. Но β-функция Геделя дает возможность закодировать последовательность $x, x^2, \ldots, x^y$ и выразить $x^y = z$ следующим предложением:

$$\exists a \exists b [\beta(a, b, 0) = 1 \wedge \forall i (i < y \to \beta(a, b, i+1) = \beta(a, b, i) \cdot x) \wedge \beta(a, b, y) = z]$$

Пришло время научиться выражать любой разрешимый предикат. Другими словами, по произвольной программе, принимающей на вход k натуральных чисел, и возвращающей «0» или «1», нужно построить арифметическое предложение, истинное, когда программа возвращает «1», и ложное, когда программа возвращает «0».

Для этой цели разумно ограничить средства нашего абстрактного языка программирования. Опишем формальный язык *Abstract C−−*. Разрешим использовать только:

1. Операторы присваивания вида `a=8;` или `a=b;`
2. Операции увеличение или уменьшения переменных на единичку: `a;` или `a++;`
3. Операторы перехода на другую строчку программы вида `goto 17;`
4. Условные конструкции вида `if (a==0) goto 226;`
5. Оператор завершения работы программы: `stop;`

Любую программу на языке *Abstract C* можно переписать на *Abstract C−−*. Понятно также, что можно ограничиться небольшим числом переменных, скажем, двумя переменными u и v (в дополнение к входным переменным). Массивы произвольной конечной длины можно кодировать натуральными числами при помощи разложения на простые множители. Например, массив `[4,1,3,3]` будет кодироваться как число $1029000 = 2^4 \cdot 3^1 \cdot 5^3 \cdot 7^3$. Договоримся, что результат работы программы — это значение переменной u, когда выполнение программы дошло до команды `stop`. Если мы

вычисляем предикат, а не функцию со значениями в натуральных числах, то можно считать, что 0 означает ложь, а любое положительное число — истину. Договоримся еще, что значение неинициализированных переменных равно 0.

Пример 6. Следующая программа вычисляет предикат $x \leqslant y$:

```
x--;
y--;
if (x==0) goto 6;
if (y==0) goto 7;
goto 1;
u=1;
stop;
```

Закодируем теперь любой разрешимый предикат некоторым арифметическим предложением. Для краткости ограничимся только одноместными предикатами. Нам понадобится понятие протокола. Протокол работы программы — это данные, описывающие состояние абстрактной вычислительной машины, выполняющей программу, на всех этапах ее работы. Другими словами, протокол — это полная история работы программы на каком-то конкретном входе Например, для машины Тьюринга протокол — это последовательность упорядоченных пар: данные на ленте и состояние машины. Для программ на языке *Abstract C−−* состояние исполняющей машины определяется:

1. Значениями всех переменных: одной входной и двух дополнительных.

2. Указанием, какая строчка в данный момент выполняется.

Поэтому протокол работы программы на языке *Abstract C−−* — это четыре последовательности: $(x_0, \ldots, x_t)$, $(u_0, \ldots, u_t)$, $(v_0, \ldots, v_t)$, $(p_0, \ldots, p_t)$, где, например, $u_1 4$ есть значение второй переменной на четырнадцатом шаге выполнения программы. Но все эти последовательности можно закодировать с помощью β-функции Геделя! А именно, рассмотрим такое предложение сигнатуры $(+, \cdot)$ с одним параметром x:

$$\exists t \exists a_x \exists b_x \exists a_u \exists b_u \exists a_v \exists b_v \exists a_p \exists b_p [\mathrm{Code}(x, t, a_x, b_x, a_u, b_u, a_v, b_v, a_p, b_p)],$$

где предложение Code(...) в скобках такое:

$$x = \beta(a_x, b_x, 0) \wedge \beta(a_u, b_u, 0) = 0 \wedge \beta(a_v, b_v, 0) = 0 \wedge \beta(a_p, b_p, 0) = 1 \wedge$$

$$\wedge \beta(a_p, b_b, t) = M \wedge \beta(a_u, b_u, t) \neq 0 \wedge \forall i[(i < t) \to \mathrm{Step}(\ldots)],$$

где $M \in \mathbb{N}$ — номер строчки, на которой программа закончила работу, а предложение

$$\mathrm{Step}(\beta(a_x, b_x, i), \ldots, \beta(a_p, b_p, i), \beta(a_x, b_x, i+1), \ldots, \beta(a_p, b_p, i+1))$$

связывает протокол работы программы с самим текстом программы. Например, для (довольно бессмысленной) программы

```
u=x;
if (u==0) goto 1;
stop;
```

предложение $\mathrm{Step}(x, u, v, p, x', u', v', p')$ будет выглядеть так:

$$[(p = 1) \to (x' = x \wedge u' = x \wedge v' = v \wedge p' = 2)] \wedge$$

$$\wedge[(p = 2 \wedge u = 0) \to (x' = x \wedge u' = u \wedge v' = v \wedge p' = 1)] \wedge$$

$$\wedge[(p = 2 \wedge u \neq 0) \to (x' = x \wedge u' = u \wedge v' = v \wedge p' = 2)] \wedge [p \neq 3].$$

Таким образом, доказана следующая теорема:

Теорема 2. *Любой разрешимый предикат* $P\colon \mathbb{N}^k \to \{0, 1\}$ *выразим в* $(\mathbb{N}, +, \cdot)$.

Следствие 1. *Любое перечислимое множество* $A \subset \mathbb{N}$ *выразимо в* $(\mathbb{N}, +, \cdot)$.

Доказательство. Пусть χ_A^* — полухарактеристическая функция множества A. Двуместный предикат «программа, реализующая χ_A^*, заканчивает работу на входе x за t шагов» разрешим. Следовательно, он выражается некоторым арифметическим предложением $Q(x, t)$. Тогда предложение $\exists t\, Q(x, t)$ выражает множество A. ■

Это утверждение играет важную роль в доказательстве первой теоремы Геделя о неполноте.

4 Что такое доказательство?

4.1 Формальные доказательства

Первая теорема Геделя о неполноте утверждает, что для любой системы аксиом арифметики существуют истинные арифметические утверждения, не доказуемые в этой системе аксиом. Но теорема Геделя — это теорема, и мы вроде как даже собираемся ее доказать. Т.е. мы собираемся доказать теорему о доказательствах! Нет ли здесь логического круга?

Дело в том, что в последнем предложении слово «доказать» употребляется в двух разных смыслах. Когда мы говорим, что докажем теорему Геделя, то имеем в виду обычное математическое доказательство, вроде, например, доказательства того, что любое разрешимое множество перечислимо. Но теорема Геделя утверждает нечто о *формальных доказательствах*. Теорема Геделя формулируется не на формальном языке арифметики, а на метаязыке. Другими словами, это метатеорема. В этом нет ничего особенно удивительного: практически все теоремы настоящего курса являются метатеоремами. Например: любой выразимый в интерпретации предикат устойчив относительно ее автоморфизмов. Или: множество доказуемых предложений перечислимо.

Чем же отличаются доказательства от формальных доказательств? Формальное доказательство — это некоторый текст, состоящий из строчек и удовлетворяющий определенным синтаксическим правилам, например:

1. Каждая строчка — замкнутое синтаксически правильное предложение некоторой сигнатуры Ω.

2. Каждая строчка является либо аксиомой теории, либо получена из предыдущих строк доказательства с помощью некоторого синтаксического правила (именно синтаксического, а не логического) из заранее фиксированного набора правил.

3. Последняя строчка текста — доказываемое предложение.

Определим строго, что такое аксиомы, теории, и какие именно синтаксические правила мы будем использовать.

Определение 5. *Теория — это произвольный набор замкнутых предложений сигнатуры Ω. Эти предложения называются аксиомами теории. Теория и система аксиом — синонимы.*

Определение 6. *Модель теории — это такая интерпретация сигнатуры Ω, в которой истинны все аксиомы этой теории. Теория называется совместной, если у нее существует хотя бы одна модель.*

Пример 7. Рассмотрим сигнатуру $(\cdot, \mathrm{inv}, e)$. Теория группы состоит из следующих аксиом. Аксиомы равенства (в любой сигнатуре, если не оговорено противное, мы предполагаем также двуместный предикатный символ $=$):

1. $\forall x(x = x)$
2. $\forall x \forall y((x = y) \to (y = x))$
3. $\forall x \forall y \forall z(((x = y) \wedge (y = z)) \to (x = z))$
4. Схема аксиом $\forall \overline{x} \forall \overline{y}(\overline{x} = \overline{y} \to \varphi(\overline{x}) = \varphi(\overline{y}))$ для любого натурального n и любого n-местного предикатного или функционального символа φ.

И, собственно, аксиомы группы:

5. $\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$
6. $\forall x(x \cdot e = x \wedge e \cdot x = x)$
7. $\forall x(x \cdot \mathrm{inv}(x) = e \wedge \mathrm{inv}(x) \cdot x = e)$

Что такое схема аксиом? Это всего лишь способ кратко записывать бесконечное число аксиом. Т.е. четвертый пункт в предыдущем списке соответствует не одной аксиоме, а бесконечно большому их числу: вместо φ можно брать любой предикатный или функциональный символ (в случае теории группы имеет смысл брать только функциональные символы). Понятно также, что приведенное бесконечное множество замкнутых предложений разрешимо. Действительно, легко написать программу, по произвольной строчке текста определяющую, является ли эта строчка аксиомой вида (4).

Для краткости договоримся включать аксиомы равенства в любую теорию, где равенство есть в сигнатуре. Тогда можно будет считать, что в теории группы всего 3 аксиомы. Впрочем, количество аксиом в теории не играет никакой роли. Например, вместо аксиом A_1, A_2, A_3 мы могли записать одну аксиому $A_1 \wedge A_2 \wedge A_3$, что никак бы не отразилось на моделях. Система аксиом группы, очевидно, совместна. Более того, у нее существует, как мы знаем, бесконечное число попарно неизоморфных моделей.

Пример 8. Рассмотрим сигнатуру $(\cdot)$ (с равенством) и систему аксиом:

1. $\forall x \forall y \forall z((x \cdot y) \cdot z = x \cdot (y \cdot z))$
2. $\exists z \forall x[(x \cdot z = x \wedge z \cdot x = x) \wedge \exists y(x \cdot y = z \wedge y \cdot x = z)]$

Получится некоторая теория, не совпадающая с предыдущей. Однако в некотором, не уточненном пока смысле она будет эквивалентна теории группы.

Пример 9. Рассмотрим сигнатуру $(0, +, \cdot, S)$, где S — одноместный функциональный символ, и систему аксиом Пеано (которую будем в дальнейшем для краткости обозначать «PA»):

1. $\forall x \neg(S(x) = 0)$
2. $\forall x \forall y(S(x) = S(y) \to x = y)$
3. $\forall x(x + 0 = x)$
4. $\forall x \forall y(x + S(y) = S(x + y))$
5. $\forall x(x \cdot 0 = 0)$
6. $\forall x \forall y(x \cdot S(y) = x \cdot y + x)$
7. Схема аксиом: $\forall \overline{y}((\varphi(0, \overline{y}) \wedge \forall x(\varphi(x, \overline{y}) \to \varphi(S(x), \overline{y}))) \to \forall x \varphi(x, \overline{y}))$

Последняя схема аксиом называется схемой математической индукции. Система аксиом Пеано, очевидно, совместна. Модель: $(\mathbb{N} \cup \{0\}, 0, +, \cdot, S)$, где S означает прибавление единицы.

Пример 10. Рассмотрим сигнатуру $(\leqslant)$ и теорию плотных линейно упорядоченных множеств без первого и последнего элементов:

1. $\forall x(x \leqslant x)$
2. $\forall x \forall y \forall z((x \leqslant y) \wedge (y \leqslant z) \to (x \leqslant z))$
3. $\forall x \forall y((x \leqslant y) \wedge (y \leqslant x) \to (x = y))$
4. $\forall x \forall y((x \leqslant y) \vee (y \leqslant x))$
5. $\forall x \exists y(y < x)$
6. $\forall x \exists y(x < y)$
7. $\forall x \forall y((x < y) \to \exists z((x < z) \wedge (z < y)))$

Запись $a < b$ можно считать сокращением для $(a \leqslant b) \wedge \neg(a = b)$.

Пример 11. Рассмотрим сигнатуру $(\in)$ и систему аксиом Цермело-Френкеля теории множеств (см. приложение). Не вполне понятно, можно ли считать «множества» моделью этой теории. Интересно также, есть ли у ZF какие-нибудь модели, в которых носитель является множеством. Например, счетным множеством.

Определение 7. *Формальное доказательство (или вывод) в теории $\mathcal{T}$ сигнатуры Ω — это конечная последовательность предложений сигнатуры Ω, каждое из которых является либо*

1) тавтологией исчисления предикатов, либо

2) аксиомой теории $\mathcal{T}$, либо

3) получено из предыдущих с помощью одного из правил вывода.

Последнее предложение этой последовательности должно быть замкнутым.

Осталось определить, что такое тавтологии и правила вывода. Тавтологии исчисления предикатов — это предложения следующих видов, где вместо A, B, C подставлены произвольные синтаксически правильные предложения, а вместо x и t — произвольные переменные (тавтологии напоминают схемы аксиом, но включаемые во все теории, $A(\frac{t}{x})$ означает предложение A, в котором все свободные вхождения переменной x заменены на t):

1. $A \to (B \to A)$
2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
3. $A \wedge B \to A$
4. $A \wedge B \to B$
5. $A \to (B \to A \wedge B)$
6. $A \to A \vee B$
7. $B \to A \vee B$
8. $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
9. $(A \to B) \to ((A \to \neg B) \to \neg A)$
10. $\neg\neg A \to A$
11. $\forall x A \to A(\frac{t}{x})$
12. $A(\frac{t}{x}) \to \exists x A$

Правила вывода — это синтаксические правила получения новых предложений по уже выведенным. Их всего три:

1. Modus Ponens: если выведены предложения A и $A \to B$, то в выводе разрешается использовать B.

2. Первое правило Бернайса: если выведено предложение $A \to B$ и x не является параметром A, то в выводе разрешается использовать $A \to \forall x B$.

3. Второе правило Бернайса: если выведено предложение $A \to B$ и x не является параметром B, то в выводе разрешается использовать $\exists x A \to B$.

В литературе правила вывода обычно записывают так:

$$\frac{A, \quad A \to B}{B}, \qquad \frac{A \to B}{A \to \forall x B}, \qquad \frac{A \to B}{\exists x A \to B}$$

Таким образом, мы полностью определили, что такое формальное доказательство. Теперь уже можно что-нибудь формально доказать!

Пример 12. Выведем в исчислении предикатов предложение вида $D \to D$, где D — произвольное замкнутое предложение какой-то сигнатуры Ω. Можно считать, что $\mathcal{T} = \emptyset$. Вот вывод:

1. $(D \to ((D \to D) \to D)) \to ((D \to (D \to D)) \to (D \to D))$ [тавтология 2 при $A = D$, $B = (D \to D)$, $C = D$];
2. $D \to ((D \to D) \to D)$ [тавтология 1];
3. $(D \to (D \to D)) \to (D \to D)$ [из (1) и (2) по правилу Modus Ponens];
4. $D \to (D \to D)$ [тавтология 1];
5. $D \to D$ [из (3) и (4) по правилу Modus Ponens].

Пример 13. Из аксиом теории группы можно вывести предложение $\forall x(\mathrm{inv}(\mathrm{inv}(x)) = x)$. Но этот вывод слишком длинный, чтобы его здесь приводить.

Пример 14. Из аксиом Пеано можно вывести известные законы для сложения и умножения, например,

$$\forall x \forall y(x + y = y + x)$$

Опять же, вывод получается слишком длинным для того, чтобы привести его на этой странице. Ясно также, что в выводе этого предложения обязательно должна использоваться схема аксиом математической индукции. Без индукции получалось бы вывести только предложения типа

$$S(S(S(0))) + S(S(0)) = S(S(0)) + S(S(S(0)))$$

Интересно, что во многих системах аксиом (для абстрактных колец, векторных пространств и т.д.) коммутативность, дистрибутивность и т.д. являются аксиомами, а в арифметике Пеано эти свойства — теоремы, причем с нетривиальным доказательством.

42. Выведите в системе аксиом Пеано предложение $S(0) + S(0) = S(S(0))$.

Как видно, вывод даже такой простой тавтологии как $D \to D$ требует некоторой писанины. Что уж говорить о более содержательных предложениях. Уровень строгости, принятый сейчас в математике, — это детский сад по сравнению с формальными доказательствами. Например, докажем (неформально), что в любой группе для любого элемента x выполнено $\mathrm{inv}(\mathrm{inv}(x)) = x$.

Доказательство. В силу ассоциативности, для любого элемента x выполнено

$$(*) \qquad \mathrm{inv}(\mathrm{inv}(x)) \cdot (\mathrm{inv}(x) \cdot x) = (\mathrm{inv}(\mathrm{inv}(x)) \cdot \mathrm{inv}(x)) \cdot x$$

Левая часть этого равенства равна $\mathrm{inv}(\mathrm{inv}(x)) \cdot e$ в силу третьей аксиомы группы, а значит, равна $\mathrm{inv}(\mathrm{inv}(x))$ в силу второй аксиомы. Аналогично, правая часть равенства $(*)$ равна x.■

Данное нами определение формального доказательства было разработано в начале XX века и приписывается, как правило, Гильберту. Есть и другие эквивалентные

определения формального доказательства (например, исчисление секвенций), дающие несколько менее длинные выводы, но все равно не сравнимые по длине с неформальными доказательствами, к которым мы привыкли. Заметим также, что нет ничего страшного в теориях с бесконечным количеством аксиом, т.к. тавтологии исчисления предикатов даже для конечных систем аксиом можно применять бесконечным числом различных способов.

43. Различные задачи на вывод в исчислении предикатов можно найти в книге Верещагина и Шеня «Языки и исчисления».

4.2 Несколько метатеорем

В настоящем разделе мы сформулируем и докажем несколько важных теорем о формальных доказательствах, т.е. метатеорем. Замкнутые предложения, выводимые из аксиом теории $\mathcal{T}$, называются теоремами теории $\mathcal{T}$. Если φ — теорема теории $\mathcal{T}$, то мы будем обозначать это так:

$$\mathcal{T} \vdash \varphi$$

Договоримся писать

$$\mathcal{T} \models \varphi,$$

если замкнутое предложение φ истинно во всех моделях теории $\mathcal{T}$. Если множество аксиом теории пусто, то $\emptyset \models \varphi$ или просто $\models \varphi$ означает по определению, что φ истинно во всех интерпретациях сигнатуры, т.е. что φ общезначимо.

К любому определению формального доказательства естественно предъявить следующее требование: все выводимые предложения должны быть истинны. Более точно, любое выводимое в теории $\mathcal{T}$ предложение должно быть истинно во всех моделях теории $\mathcal{T}$. Это достаточно очевидное утверждение называется теоремой корректности исчисления предикатов:

Утверждение 9. *Если $\mathcal{T} \vdash \varphi$, то $\mathcal{T} \models \varphi$.*

Например, если бы включили в набор тавтологий исчисления предикатов схему $A \to A \wedge B$, то полученное исчисление уже не было бы корректным. Бывают, правда, теории, в которых можно вывести все, что угодно.

Определение 8. *Теория $\mathcal{T}$ называется противоречивой, если существует такое замкнутое предложение φ, для которого $\mathcal{T} \vdash \varphi$ и $\mathcal{T} \vdash \neg\varphi$.*

Утверждение 10. *В противоречивой теории любое замкнутое предложение выводимо.*

Доказательство. Тривиально: нужно использовать тавтологию (9) и правило Modus Ponens. ■

Это утверждение можно пояснить более наглядным образом. Как, например, из предложений «сейчас 2011 год» и «сейчас 2012 год» вывести, что Земля плоская?

Очень просто. Мы лишь ослабим предложение «сейчас 2011 год», если заменим его на «либо сейчас 2011 год, либо Земля плоская». Но ведь сейчас 2012 год. Следовательно, Земля плоская.

Нам понадобится также одна полезная лемма, связывающая выводимость предложения φ в теории $\mathcal{T}$ с выводимостью предложения «если $\mathcal{T}$, то φ» в пустой теории. Пусть A — некоторое замкнутое предложение.

Лемма 2. *$\mathcal{T} \vdash (A \to \varphi)$ тогда и только тогда, когда $\mathcal{T} \cup \{A\} \vdash \varphi$.*

Эта лемма называется леммой о дедукции. В одну сторону она очевидна: если из аксиом $\mathcal{T}$ выводится предложение $A \to \varphi$, то из $\mathcal{T} \cup \{A\}$ тоже можно вывести $A \to \varphi$, а затем, пользуясь правилом Modus Ponens, вывести A. В обратную сторону она менее очевидна: можно указать процедуру, как по формальному доказательству в теории $\mathcal{T} \cup \{A\}$ построить формальное доказательство в теории $\mathcal{T}$, но утверждения $A \to \varphi$. Если теория $\mathcal{T} = \{A_1, \ldots, A_n\}$ конечна, то лемма о дедукции фактически утверждает эквивалентность выводимостей $\mathcal{T} \vdash \varphi$ и $\vdash (A_1 \wedge \ldots \wedge A_n) \to \varphi$.

Из теоремы корректности легко следует следующая метатеорема:

Следствие 2. *Если теория $\mathcal{T}$ совместна, то она непротиворечива.*

Доказательство. От противного: предположим, что $\mathcal{T}$ противоречива. Тогда существует предложение φ, для которого $\mathcal{T} \vdash \varphi$ и $\mathcal{T} \vdash \neg\varphi$. По теореме о корректности отсюда следует, что $\mathcal{T} \models \varphi$ и $\mathcal{T} \models \neg\varphi$. Но у $\mathcal{T}$ есть как минимум одна модель $\mathcal{M}$. Значит, оба предложения φ и $\neg\varphi$ истинны в $\mathcal{M}$. Противоречие. ■

Итак, теорема о корректности утверждает, что любое выводимое в теории $\mathcal{T}$ предложение истинно во всех моделях этой теории. В 1929 году Гедель доказал, что верно и обратное: любое предложение, истинное во всех моделях теории $\mathcal{T}$, выводимо в $\mathcal{T}$! Это утверждение называется теоремой о полноте исчисления предикатов. Если, скажем, исключить из набора тавтологий исчисления предикатов какую-нибудь одну, то полученное определение формального доказательства уже не будет удовлетворять свойству полноты.

Как можно доказать теорему о полноте? Как использовать истинность во всех моделях. Оказывается, полнота исчисления предикатов следует из следующей теоремы:

Теорема 3. *Любая непротиворечивая теория $\mathcal{T}$ имеет модель.*

Следствие 3. *Если $\mathcal{T} \models \varphi$, то $\mathcal{T} \vdash \varphi$.*

Доказательство. От противного: предположим, что предложение φ невыводимо в $\mathcal{T}$. Тогда теория $\mathcal{T} \cup \{\neg\varphi\}$ непротиворечива (т.к. если бы она была противоречива, то в ней было бы выводимо любое предложение, в том числе φ). Следовательно, по теореме (3), теория $\mathcal{T} \cup \{\neg\varphi\}$ имеет модель $\mathcal{M}$. Тогда $\neg\varphi$ должно быть истинно в этой модели, а φ — ложно. Противоречие.■

Теорему же (3) можно доказать, явно построив модель по сигнатуре и теории.

Определение 9. *Теория называется полной, если для любого замкнутого предложения из этой теории выводимо либо оно само, либо его отрицание.*

Полнота какой-либо теории и полнота исчисления предикатов — это разные понятия, обозначенные одним и тем же словом. Не путайте их!

Наконец, настало время соединить теорию алгоритмов с наукой о формальных доказательствах, моделях, выразимых предикатах и т.д..

Определение 10. *Теория называется разрешимо аксиоматизируемой, если множество ее аксиом разрешимо.*

Теории, не являющиеся разрешимо аксиоматизируемыми, — это что-то очень экзотическое. Представьте: не существует алгоритма, определяющего, является ли данное предложение аксиомой. Поэтому после настоящего раздела под теориями мы будем подразумевать именно разрешимо аксиоматизируемые теории. Все рассматриваемые до сих пор теории были такими. Очевидно (из определения формального доказательства) следующее утверждение:

Утверждение 11. *Множество доказательств в разрешимо аксиоматизируемой теории разрешимо.*

Другими совами, задача определения по произвольному тексту, является ли он синтаксически правильным формальным доказательством, алгоритмически разрешима. Правильность формального доказательства может быть проверена механически, без участия человека.

Следствие 4. *Множество теорем разрешимо аксиоматизируемой теории перечислимо.*

Доказательство. Будем перебирать все тексты (например, упорядочив их по длине). Для каждого текста можно алгоритмически определить, является ли этот текст формальным доказательством, и, если да, то какое предложение он доказывает. Доказанные предложение будем подавать на выход программы. Любое доказуемое предложение рано или поздно окажется в выходной последовательности.■

Какая хорошая теорема! Существует программа, генерирующая все доказуемые предложения. Да, эта программа не сможет определить, доказуемо ли данное конкретное предложение (т.к. не будет знать, когда ей остановиться и выдать, что доказательства нет), но перечислимость это тоже неплохо. Мы можем ее запустить и смотреть, как она постепенно развивает науку, доказывая все новые и новые предложения и не пропуская ни одного. К сожалению, такая программа бесполезна с практической точки зрения сразу по нескольким причинам:

1. Она будет работать слишком медленно. Количество правильных доказательств экспоненциально возрастает с длиной текста, а формальные доказательства, как мы знаем, бывают *очень* длинными.

2. Среди всех доказуемых предложений *интересные* предложения образуют лишь ничтожную часть. Почти все предложения, которые будет выдавать перечисляющая программа, будут мусором: они будут, конечно, верны, но не будут интересны для человека.

3. Можно частично решить предыдущую проблему, если исправить программу так, чтобы она выводила только относительно короткие предложения. Тогда мы, может быть, узнаем несколько новых математических истин. Но какой толк в знании того, что какое-то предложение является истинным, если мы не понимаем его доказательства? Представьте себе огромное формальное доказательство, где каждый отдельный переход проверить можно, но все доказательство в целом, его идея, остаются совершенно непонятными.

44. Докажите, что множество теорем полной (и разрешимо аксиоматизируемой) теории разрешимо.

4.3 Неформальные доказательства

Предыдущие разделы должны были убедить читателя в том, что доказательства, встречающиеся в математических статьях и книгах, — это ни в коем случае не формальные доказательства. Действительно, их авторы не оговаривают сигнатуру языка в начале статьи, как правило не фиксируют аксиомы, которыми они будут пользоваться, а если даже и фиксируют, то в самих доказательствах апеллируют к логике и говорят, что такие-то и такие-то утверждения являются истинными.

Сейчас в математике доказательства играют примерно ту же роль, что они играли у Евклида: доказательство — это надежный способ установления истинности какого-либо утверждения. Аксиоматический метод позволяет устанавливать истинность или ложность большого количества утверждений, исходя лишь из нескольких базовых и самоочевидных утверждений, называемых аксиомами. Часто системы аксиом используются в качестве определений (как, например, аксиомы группы).

Аксиомы же в формальных доказательствах — это строки, конечные последовательности букв. Они ничего не утверждают и ничего не определяют. Рассуждать об их истинности или очевидности бессмысленно. Из этих строк по определенным правилам можно получать новые строки. Эти новые строки называются теоремами. При таком подходе неправильно говорить, например, что прямая — это то, что удовлетворяет аксиомам. В формальных доказательствах нет никакого «то»! Есть только аксиомы. Это последовательности букв. Нельзя «удовлетворять аксиомам»!

Не будет преувеличением сказать, что математическое доказательство — это текст на естественном языке, убеждающий читателя в том, что доказываемое утверждение истинно. Можно еще сделать одно уточнение: только с помощью этого текста читатель потом сможет убеждать других людей. В XIX веке людей уже не убеждали те же тексты, что считались доказательствами в XVIII веке. Наоборот, в наше время немногие бы стали сомневаться в доказательстве гипотезы Гольдбаха, использующем

трансфинитную индукцию и ультрафильтры. Формальное же доказательство — это конкретный математический объект, вроде группы или треугольника.

4.4 Формализм как программа обоснования математики

Понятие математической истины довольно туманное. Что именно мы подразумеваем, когда говорим, например, что число π иррационально? Чему этот факт соответствует в наблюдаемой физической реальности? Ведь действительных чисел на самом деле нет. Или они все же существуют в каком-то особом, идеальном мире?

Уместно спросить, а как вообще в математике устанавливается истинность? Уже очень давно истинность математических утверждений устанавливают с помощью доказательств. В конце XIX века уточнение и формализация понятия математического доказательства привели к заманчивой возможности считать, что истинные утверждения — это, по определению, утверждения, доказуемые в некоторой системе аксиом. Такой подход к математической истине обычно связывают с именем Гильберта и называют формализмом. К формализму располагало и недавнее открытие гиперболической геометрии: было понято, что пятый постулат Евклида не является истинным или ложным в каком-то абсолютном смысле. Все, что мы можем сказать о пятом постулате, — это то, что ни он, ни его отрицание не выводятся из остальных аксиом.

На пути успешного проведения формализма есть, однако, одно препятствие. Определять истинные утверждения как доказуемые в некоторой системе аксиом бессмысленно, если эта система аксиом противоречива (а значит, любое утверждение в ней является доказуемым). Конечно, в непротиворечивости таких систем как аксиомы Пеано или аксиомы группы никто не сомневается. Но математика — это не только арифметика. В математике много областей, и в каждой из них могут возникнуть свои системы аксиом, непротиворечивость которых будет совсем не очевидна.

К счастью, изучать все эти системы аксиом совершенно не обязательно. К концу XIX века было осознано, что все многообразие математических объектов и отношений можно свести к очень малому числу базовых объектов и отношений. А именно, к единственному типу объектов — множествам, и единственному отношению — отношению принадлежности. Функции, натуральные числа, группы, топологические пространства — это все множества! Например, функция из множества A в множество B — это подмножество прямого произведения $A \times B$, удовлетворяющее определенным условиям, а натуральные числа (с нулем) можно отождествить с множествами $\{\emptyset\}$, $\{\emptyset, \{\emptyset\}\}$, $\{\emptyset, \{\emptyset, \{\emptyset\}\}\}$, ... Была также придумана система аксиом для теории множеств (получившая название системы аксиом Цермело-Френкеля или ZF), к которой удавалось сводить все известные математические доказательства.

Правда, непротиворечивость ZF или других систем аксиом теории множеств совершенно не очевидна! Действительно, не получается привести достаточно надежную модель ZF. Парадоксы Рассела и других научили математиков с осторожностью относиться к таким общим и абстрактным объектам как множества. Да, в ZF получить известные парадоксы наивной теории множеств было уже нельзя. Но где гарантия, что невозможны какие-нибудь другие противоречия?

Необходимо было доказать непротиворечивость ZF, причем доказать надежным способом. Почему бы и нет? Ведь непротиворечивость любой системы аксиом — это утверждение о том, что, оперируя по определенным комбинаторным правилам с конечными строками, нельзя получить строку «$\perp$». Это хорошая олимпиадная задачка. Теоретически, непротиворечивость любой системы аксиом можно доказать финитными методами. Например, можно попробовать доказать непротиворечивость ZF в аксиоматике Пеано. Если это получится, то математика будет полностью обоснована с позиции формализма! Действительно, можно забыть про истинность утверждений и оставить только понятие доказуемости. Непротиворечивость же единственной используемой системы аксиом будет обоснована надежным методом.

В математике невозможно определить все: можно определить какие-то понятия через более фундаментальные, те, в свою очередь, — через еще более фундаментальные и т.д.. Но где-то мы должны будем остановиться, дойдя до базовых неопределяемых понятий. Формализм не в коем случае не ставит цели каким-либо образом преодолеть эту ситуацию. Формализм лишь требует, чтобы эти базовые понятия были надежными: не как понятие наивного множества, а как понятия аксиом, правил вывода и т.д..

В 1931 году Гедель доказал, что даже непротиворечивость PA не доказуема в PA. Тем более непротиворечивость ZF не доказуема в PA. Это утверждение называется второй теоремой Геделя о неполноте.

Читателю уместно спросить, как можно рассчитывать на доказательство непротиворечивости какой-либо системы аксиом в PA, если аксиомы Пеано позволяют доказывать только утверждения о натуральных числах, типа коммутативности умножения или теоремы Ферма? Дело в том, что утверждения о непротиворечивости системы аксиом или о ее полноте можно определенным способом закодировать на арифметическом языке. Подробнее об этом написано в следующих разделах.

Вторая теорема Геделя о неполноте очень просто следует из первой. Первую же можно сформулировать как минимум в двух вариантах: синтаксическом и семантическом. Синтаксический вариант первой теоремы Геделя о неполноте утверждает, что любая достаточно богатая (охватывающей арифметику) непротиворечивая система аксиом обязательно неполна. Неполнота системы аксиом неприятна для формализма, но не страшна: вполне можно смириться с существованием недоказуемых и неопровержимых утверждений. Семантический же вариант утверждает, что для любой достаточно богатой непротиворечивой системы аксиом существует предложение, не доказуемое в этой системе аксиом, но истинность которого мы должны признать, если считаем эту систему аксиом непротиворечивой. Другими словами, для любого определения формального доказательства существует такое недоказуемое утверждение, истинность которого мы обязаны признать, если считаем наше определение формального доказательства корректным. В семантическом варианте первая теорема Геделя практически совпадает со второй и гораздо более опасна для формализма.

5 Теоремы Геделя о неполноте

5.1 Насколько хороша истина?

Первая теорема Геделя о неполноте утверждает, что не существует такой непротиворечивой системы аксиом, в которой доказуемы все истинные в $(\mathbb{N}, +, \cdot)$ предложения. Введем теперь необходимые обозначения.

Как обычно, занумеруем каким-нибудь разрешимым образом все замкнутые предложения сигнатуры $(+, \cdot)$. Будет удобно, хотя это совершенно не обязательно, ввести такую нумерацию, при которой более коротким предложениям соответствуют меньшие номера. Нумерация задает раскраску натурального ряда в два цвета: цветам соответствует истинность или ложность соответствующих предложений. Пусть $T \subset \mathbb{N}$ — множество номеров истинных предложений.

45. Плотность каких предложений будет большей, истинных или ложных?

Фиксируем теперь какую-нибудь разрешимую систему аксиом Γ сигнатуры $(+, \cdot)$, для которой $(\mathbb{N}, +, \cdot)$ является моделью. Например, можно взять аксиоматику Пеано (если выразить S и 0 через $+$). Пусть $D \subset \mathbb{N}$ — множество номеров предложений, доказуемых в Γ. По теореме о корректности исчисления предикатов $D \subset T$.

Теорема 4. *(первая теорема Геделя о неполноте, семантическая версия)* $D \neq T$.

Сравним сразу эту теорему с теоремой о полноте исчисления предикатов. Теорема Геделя о полноте утверждает, что если предложение истинно *во всех* моделях системы аксиом Γ, то оно доказуемо в Γ. Теорема Геделя о неполноте утверждает, что, какую бы мы систему аксиом Γ не взяли, из истинности предложения в $\mathbb{N}$ (т.е. в одной конкретной модели) не следует доказуемость. Другими словами, сущность натуральных чисел невозможно схватить аксиоматически. Комбинируя теорему о полноте с теоремой о неполноте, мы приходим к выводу, что у системы аксиом Пеано, например, есть модели, не изоморфные $(\mathbb{N}, +, \cdot)$. Такие модели называются нестандартными.

Доказательство первой теоремы Геделя может быть разбито на несколько шагов.

Теорема 5. *Множество $T \subset \mathbb{N}$ предложений, истинных в $(\mathbb{N}, +, \cdot)$*
1) неразрешимо;
2) неперечислимо;
3) невыразимо.

Где под выразимостью сейчас и в дальнейшем будем подразумеваться именно арифметическую выразимость, т.е. выразимость в $(\mathbb{N}, +, \cdot)$. Пункт (3) называется теоремой Тарского о невыразимости истины. Как мы знаем, из невыразимости следует неперечислимость, а из неперечислимости — неразрешимость. Пункт (2), фактически, и есть первая теорема Геделя о неполноте, причем в наиболее прямой и приятной для математика формулировке (правда, требующей знания теории алгоритмов). Действительно, для любого разумного определения формального доказательства множество

доказуемых предложений будет перечислимо. Т.е. теорема Геделя сразу следует из (2) или (3) и уже известного нам утверждения:

Теорема 6. *Множество $D \subset \mathbb{N}$ предложений, доказуемых в Γ*
1) перечислимо;
2) выразимо.

Разрешимость множества D зависит, конечно, от конкретной системы аксиом.

∘ **46.** Докажите, что задача проверки общезначимости предложения языка первого порядка, алгоритмически неразрешима для сигнатуры, состоящей из одного двуместного предикатного символа и одного двуместного функционального символа.

Полезно также доказать, что существуют выразимые, но неперечислимые множества:

Теорема 7. *Множество $\mathbb{N}\backslash S$ неостанавливающихся программ*
1) неразрешимо;
2) неперечислимо;
3) выразимо.

Доказательство. Неперечислимость $\mathbb{N}\backslash S$ была доказана ранее. Также была доказана перечислимость, и следовательно, выразимость S. Выразимость $\mathbb{N}\backslash S$ следует из свойств выразимых множеств. Действительно, если множество S выражается предложением $P(x)$, то множество $\mathbb{N}\backslash S$ выражается предложением $\neg P(x)$.■

Заметим еще, что из семантической версии теоремы Геделя легко следует версия синтаксическая:

Теорема 8. *Пусть Γ — система аксиом сигнатуры $(+,\cdot)$, для которой $(\mathbb{N},+,\cdot)$ является моделью. Тогда Γ неполна.*

5.2 Теорема Тарского о невыразимости истины

В доказательстве теоремы Тарского нам понадобится (разрешимая) нумерация множества предложений с одним параметром. Пусть $F_n(\cdot)$ — предложение с одним параметром, соответствующее $n \in \mathbb{N}$.

Теорема 9. *Множество $T \subset \mathbb{N}$ предложений, истинных в $(\mathbb{N},+,\cdot)$, невыразимо в $(\mathbb{N},+,\cdot)$.*

Доказательство. Будем доказывать от противного: предположим, что множество T выражается некоторым предложением $\tau(\cdot)$. Рассмотрим тогда такое предложение $P(x)$:

$$\exists z(\neg\tau(z) \wedge \mathrm{Subst}(z,x,x)),$$

где $\mathrm{Subst}(p,q,r)$ выражает предикат: p есть номер предложения (в нумерации замкнутых предложений), которое получится, если в q-ое предложение с одним параметром

подставить переменную r вместо этого параметра. Другими словами, $\mathrm{Subst}(p, q, r)$ выражает предикат: p есть номер $F_q(r)$. Очевидно, этот предикат разрешим, и поэтому такое предложение $\mathrm{Subst}(p, q, r)$ действительно существует. При наличии терпения его можно даже явно выписать как предложение в сигнатуре $(+, \cdot)$.

Но $P(x)$ — предложение с одним параметром. Следовательно, для некоторого натурального N

$$P(x) \equiv F_N(x)$$

Зададимся теперь вопросом, истинно или ложно замкнутое предложение $P(N)$. Если $P(N)$ истинно, то существует такое натуральное число Z, что замкнутое предложение, имеющее номер Z, ложно и при этом Z есть номер предложения $F_N(N) \equiv P(N)$! Если же $P(N)$ ложно, то получаем аналогичное противоречие. Следовательно, исходное предположение о выразимости T было неверным.■

Тянет сказать, что предложение $P(N)$ утверждает собственную ложность. На наш взгляд, это неверно. $P(N)$ не утверждает собственную ложность подобно тому, как предложение «это предложение ложно» утверждает собственную ложность. $P(N)$ — это обычное, хотя и очень длинное, арифметическое предложение с множеством скобок, кванторов, переменных и знаков $=$, $+$, $\cdot$ (конечно, предложения $P(N)$ не существует, как мы только что доказали). Оно утверждает существование некоторого $Z \in \mathbb{N}$ с какими-то очень странными арифметическими свойствами. Лишь из метарассуждений мы знаем, что $P(N)$ в некотором смысле кодирует парадокс лжеца. Другими словами, предложение на формальном языке арифметики никак не может утверждать собственную ложность — предложение «это предложение ложно» построимо лишь в сверхбогатых языках типа русского, где возможно смешение уровня и метауровня. В доказательстве же теоремы Тарского (и теорем Геделя) такого смешения не происходит.

Таким образом, первая теорема Геделя о неполноте полностью доказана. Но мы на этом не остановимся и приведем еще несколько полезных для математики и интересных для философии рассуждений. Оказывается, небольшое изменение доказательства теоремы Тарского позволяет *явно* построить истинное, но не доказуемое предложение.

Пусть предложение $\mathrm{Provable}(\cdot)$ выражает множество доказуемых предложений D. Рассмотрим предложение $P(x)$:

$$\exists z(\neg \mathrm{Provable}(z) \wedge \mathrm{Subst}(z, x, x))$$

Из рассуждений, аналогичных рассуждениям в доказательстве теоремы Тарского следует, что $P(N)$ кодирует собственную недоказуемость! Может ли $P(N)$ быть ложным? Нет, т.к. тогда $P(N)$ доказуемо, и поэтому истинно. Может ли $P(N)$ быть доказуемым? Нет, т.к. тогда оно истинно, и поэтому недоказуемо. Следовательно, $P(N)$ — истинное, но не доказуемое предложение! При наличии терпения предложение $P(N)$ можно выписать явно.

Итак, по системе аксиом Γ мы явно предъявили истинное, но не доказуемое в Γ предложение. Уместно спросить: откуда же мы знаем, что оно истинно, если оно не

доказуемо? Ответ: из метарассуждений. Более точно, в доказательстве истинности $P(N)$ мы использовали, что любое формально доказуемое предложение истинно, т.е. непротиворечивость системы аксиом Γ.

Можно добавить предложение $P(N)$ в Γ и получить новую систему аксиом $\Gamma_1 = \Gamma \cup \{P(N)\}$. Но к Γ_1 рассуждение Геделя тоже будет применимо: найдется предложение $P_1(N_1)$, не доказуемое в Γ_1, но истинность которого мы обязаны признать, т.к. Γ_1 непротиворечива, и т.д.. Здесь хорошо заметна параллель с проблемой остановки. По любому частичному решателю проблемы остановки мы строили программу, с которой этот решатель не справлялся, но из метарассуждений следовало, что эта программа не остановится. Тогда можно было улучшить решатель, добавив в его базу знаний эту программу и получить новый. Для этого нового решателя тоже существует подобная программа и т.д..

Все эти рассуждения, конечно, проходят, если в качестве Γ взять систему аксиом Пеано. Можно явно выписать истинное предложение, не доказуемое в PA. Оно, правда, будет очень длинным и не интересным с точки зрения арифметики. Однако сейчас уже известны относительно короткие и интересные арифметические истины, не доказуемые в PA. Один из таких примеров — знаменитая теорема Гудстейна.

5.3 Еще одно доказательство теоремы Геделя

Развитая нами теория позволяет дать еще одно, более короткое доказательство первой теоремы Геделя о неполноте. Краткость будет достигнута за счет того, что мы будем использовать выразимость всех перечислимых множеств, а не только множества D доказуемых предложений. Основная идея доказательства: если множество T истинных арифметических предложений перечислимо, то перечислимо и любое выразимое множество.

Теорема 10. *Множество T истинных арифметических предложений неперечислимо.*

Доказательство. От противного: пусть T перечислимо. Перечислим тогда произвольное выразимое множество $A \subset \mathbb{N}$. Пусть A выражается предложением $P(x)$. Запустим программу, перечисляющую T. Если эта программа выдает предложение $P(S \dots S0)$, где S написано k раз, то $k \in A$. Таким образом, можно построить программу, перечисляющую A. Но бывают и выразимые неперечислимые множества. Противоречие.■

5.4 Вторая теорема Геделя о неполноте

Вторая теорема Геделя о неполноте утверждает, что непротиворечивость системы аксиом Пеано не доказуема в ней самой (и следовательно, непротиворечивость ZF не доказуема в PA). На уровне идей вторая теорема Геделя элементарно следует из первой. Действительно, первая теорема утверждает, что существует предложение истинное G, не доказуемое в PA, но доказуемое, если учитывать еще и непротиворечивость

PA. Поэтому, если бы непротиворечивость PA была бы доказуема в PA, то G было бы тоже доказуемо в PA. Постараемся теперь провести это рассуждение аккуратно.

Для начала, как непротиворечивость чего угодно в принципе может быть доказана в PA, если в PA доказываются только арифметические предложения? Дело в том, что в PA можно попытаться доказать предложение, в некотором смысле кодирующее непротиворечивость PA. А именно, пусть φ — замкнутое предложение сигнатуры $(+, \cdot)$. Предикат «X есть номер доказательства в PA предложения φ» разрешим. Следовательно, он выражается некоторым предложением $\mathrm{Proof}_\varphi(x)$. Предложение

$$\exists x \mathrm{Proof}_\varphi(x)$$

кратко будем записывать как $\Box\varphi$. Таким образом, по любому замкнутому предложению φ мы построили другое замкнутое предложение, кодирующее доказуемость φ. Точнее, из метарассуждений мы знаем, что $\Box\varphi$ истинно тогда и только тогда, когда φ доказуемо. Предложение

$$\neg\Box\bot$$

кодирует непротиворечивость PA.

Важно понимать, что $\Box\varphi$ — это именно арифметическое предложение, т.е. предложение той же сигнатуры $(+, \cdot)$. Оно, конечно, очень длинное и с множеством кванторов. В доказательстве второй теоремы Геделя очень важно не путать два уровня: уровень и метауровень. Договоримся поэтому арифметические предложения записывать странными значками, латинскими или греческим буквами, а утверждения метауровня — на русском. Например, $\Box\varphi$ — это арифметическое предложения, а «φ доказуемо» — утверждение метаязыка.

Лемма 3. *Пусть G — истинное, но не доказуемое в PA предложение, полученное применением к PA конструкции из доказательства первой теоремы Геделя. Тогда предложение $\neg\Box\bot \to G$ доказуемо в PA.*

При наличии желания и терпения можно предъявить это формальное доказательство явно. Мы, однако, не будем его предъявлять, т.к. эта лемма очень ожидаема: фактически она утверждает, что финитная часть первой теоремы Геделя (если PA непротиворечива, то G истинно), может быть формально доказана в PA. Другими словами, аксиоматика Пеано достаточно богата для того, чтобы провести в ней финитную часть доказательства первой теоремы. Для лучшего понимания сформулируем еще одно естественное утверждение, которое, правда, не понадобится в доказательстве второй теоремы Геделя.

Утверждение 12. *Если предложение $\Box\varphi$ истинно в стандартной модели PA, то $\Box\varphi$ доказуемо в PA.*

Теорема 11. *(вторая теорема Геделя о неполноте) Предложение $\neg\Box\bot$ недоказуемо в PA.*

Доказательство. От противного: пусть предложение $\neg\Box\bot$ доказуемо в PA. По лемме, в PA доказуемо и предложение $\neg\Box\bot \to G$. Тогда, применяя правило Modus Ponens, можно получить и формальное доказательство G в PA. Противоречие.■

Аналогично получается вторая теорема Геделя для ZF ($\neg\Box\bot$ теперь является предложением сигнатуры $\{\in\}$):

Теорема 12. *Если ZF непротиворечива, то предложение $\neg\Box\bot$ недоказуемо в ZF.*

Мы получили забавный факт: если система аксиом ZF непротиворечива, то ее непротиворечивость недоказуема, а если ZF противоречива, то ее непротиворечивость доказуема. Таким образом, если у кого-то вдруг получится доказать непротиворечивость ZF, то мы точно будем знать, что ZF противоречива.

Внимательный читатель заметил, что первая теорема Геделя была доказана для произвольной системы аксиом сигнатуры $(+,\cdot)$ (для которой натуральные числа являются моделью), а вторая — именно для PA. Насколько существенна именно аксиоматика Пеано в доказательстве второй теоремы? Ответ: она довольно существенна, т.к. в более слабой системе аксиом может не получиться доказать предложение

$$\neg\Box\bot \to G$$

В начале XXI века были открыты системы аксиом арифметики, которые позволяют доказать собственную непротиворечивость. В этих аксиомах доказательство первой теоремы Геделя, конечно, не может быть проведено.

Вторая теорема Геделя дает еще один метод (эквивалентный первому, как мы доказали) постепенного расширения системы аксиом. Обозначим систему аксиом PA через Γ_0 и образуем $\Gamma_1 = \Gamma_0 \cup \chi_0$, где χ_0 кодирует непротиворечивость Γ_0. Аналогично будем образовывать Γ_2, Γ_3 и т.д..

47. Доказуема ли первая теорема Геделя о неполноте в системе аксиом Пеано?

Решение: очевидно, нет. Как и многие другие метатеоремы, она даже не формулируется на языке арифметики! Тем не менее, в системе аксиом Пеано доказуемы предложения

$$\neg\Box\bot \to \neg\Box G \qquad \text{и} \qquad \neg\Box G \leftrightarrow G,$$

где G — построенное ранее Геделевское предложение. В некотором смысле эти арифметические предложения кодируют первую теорему Геделя о неполноте.

6 Злоупотребления теоремой Геделя о неполноте

6.1 Стандартные злоупотребления

Первая теорема Геделя о неполноте является, пожалуй, самой известной теоремой математики XX века. Эта известность, вместе с ее большим философским значением и востребованностью среди нематематиков породили множество неточных или даже неверных формулировок теоремы Геделя. Например:

1. Любая формальная система либо противоречива, либо неполна.

2. Существуют недоказуемые истины.

3. Существуют утверждения, которые нельзя ни доказать, ни опровергнуть.

4. Истинность предложений в языке средствами этого языка невыразима (неточная формулировка теоремы Тарского о невыразимости истины). Или просто: истина невыразима.

5. Непротиворечивость системы аксиом не может быть доказана в самой этой системе аксиом.

Утверждение (1), даже если уточнить, что такое формальная система, просто неверно. В математике известно много систем аксиом, являющихся одновременно непротиворечивыми и полными. Среди них встречаются и нетривиальные, например, система аксиом Тарского евклидовой геометрии. Говорить, что из теоремы Геделя следует, что любое законодательство либо противоречиво, либо неполно, можно только ради шутки. Законодательства, конечно, не являются формальными системами. Формулировка (2) уже лучше: необходимо только уточнить, что имеется в виду истинность в интерпретации $(\mathbb{N}, +, \cdot)$ и что такое доказуемость.

Утверждение (3) верно для любой непротиворечивой системы аксиом сигнатуры $(+, \cdot)$, для которой $(\mathbb{N}, +, \cdot)$ является моделью, или же для более общих систем типа ZF в случае их непротиворечивости. Формулировка (4) также становится точной, если заменить «язык» на «язык первого порядка с сигнатурой $(+, \cdot, \ldots)$» и под выразимостью подразумевать выразимость в $(\mathbb{N}, +, \cdot, \ldots)$. (5) неверно сразу по двум причинам: во-первых, предложения о противоречивости или непротиворечивости системы аксиом не могут быть даже сформулированы на языке системы, и во-вторых, бывают аксиоматики, в которых предложение, кодирующее их непротиворечивость, таки может быть доказано. Правильная формулировка второй теоремы Геделя такова: предложение, кодирующее непротиворечивость ZF, недоказуемо в PA. Или: предложение, кодирующее непротиворечивость PA, недоказуемо в PA.

Учитывая существование теоремы Геделя о полноте, полных разрешимых теорий, самоверифицируемых теорий, арифметики второго порядка и т.д., очень важно помнить и понимать строгую формулировку теорем о неполноте. Лучше всего, как нам кажется, запомнить такую формулировку: множество истинных арифметических предложений невыразимо, и следовательно, неперечислимо. Помня о том, что такое перечислимое множество и почему при любом разумном подходе к определению формального доказательства множество доказуемых предложений обязательно перечислимо, из нее легко вывести более значимые для философии формулировки. Важно также помнить идею доказательства первой теоремы о неполноте (явное построение предложения, кодирующего собственную недоказуемость).

Однако теорему Геделя можно не только неверно сформулировать, но и некорректно интерпретировать. Неправильно, например, утверждать, что теорема Геделя

указывает на принципиальную ограниченность возможностей математических доказательств и что теперь нам можно все доказывать нестрого. Теорема Геделя указывает на принципиальную ограниченность возможностей *формальных* доказательств. Доказательства, реально используемые в математике, как мы говорили, — это ни в коем случае не формальные доказательства. Формальные доказательства являются математическими объектами, наподобие треугольников или групп. Математические же доказательства находятся уже на метауровне. В этом смысле сама теорема Геделя является обыкновенной математической теоремой, не уступающей по строгости формулировки и доказательства, например, большой теореме Ферма. Более того, арифметическое предложение, кодирующее первую теорему Геделя, может быть даже формально доказано в системе аксиом Пеано.

То, что математическая истина выходит за рамки любого формализма, ясно и без теоремы Геделя. Действительно, как бы мы иначе решали, какие аксиомы и правила вывода брать для построения формальной системы? Теорема Геделя говорит большее: невозможно обратиться к понятию истинности только один раз, в самом начале, а потом уже строить математику на основе формальных синтаксических правил.

6.2 Искусственный интеллект и искусственное сознание

В книге Роджера Пенроуза «Новый ум короля. О компьютер, мышлении и законах физики» теорема Геделя возникает в другом контексте.

Сейчас нас повсюду окружают компьютеры и компьютерные программы. Программы решают дифференциальные уравнения, играют в шахматы лучше человека и даже создают и изменяют другие программы. Могут ли компьютерные технологии в будущем развиться настолько, что программы уже ничем не будут уступать человеку? Настолько, что любой вид интеллектуальной деятельности, которой занимаются люди, будет также доступен компьютерам? Другими словами, возможно ли создание искусственного интеллекта, не уступающего по своим возможностям человеческому интеллекту (такой искусственный интеллект в этом разделе мы будем называть сильным ИИ)? Наконец, не являемся ли мы сами всего лишь компьютерами, «сделанными из мяса»?

Существование алгоритмически неразрешимых задач доказывает лишь невозможность абсолютного ИИ, решающего любую заданную ему задачу. Но нам и не нужен абсолютный интеллект. Человек же не может гарантировать, что сможет решить проблему остановки для произвольной входной программы. То же относится и к первой теореме Геделя о неполноте, которую интерпретировать так: машине (формальной системе) невозможно объяснить, что такое истина. Но знаем ли мы сами, что такое истина?

Более серьезные препятствия для сильного ИИ — это само доказательство теоремы Геделя и рассуждение о частичном решателе проблемы остановки. Действительно, мы увидели, что все алгоритмические системы-решатели характеризуются следующим: внешний по отношению к системе наблюдатель легко сможет эту систему превзойти. А по отношению к компьютеру человек легко может занять позицию внешнего наблю-

дателя. Правда, этот аргумент не годится, если считать, что по отношению к любому человеку тоже можно занять позицию внешнего наблюдателя.

К вопросу о возможности сильного ИИ можно подойти с теоретических позиций, уточняя, что такое интеллект, ум, мышление, а можно придумать какой-нибудь хороший практический критерий интеллектуальности. Такой критерий описал Алан Тьюринг в 1950 году. Тьюринг задался вопросом: как мы узнаем, что окружающие люди интеллектуальны? Конечно, из беседы с ними. Поэтому программу, умеющую вести осмысленные разговоры не хуже человека следует признать интеллектуальной.

Самый простой вариант теста Тьюринга можно описать так: перед человеком-судьей находятся два терминала, позволяющие обмениваться текстовыми предложениями с собеседниками, находящимися в двух других комнатах. Известно, что один из собеседников является человеком, а другой — программой. Задача судьи — определить, кто из собеседников человек, а кто — программа. При этом и у собеседника-человека и у программы есть цель убедить судью в своей человечности. Если судья ошибается в своем решении, то считается, что программа прошла тест Тьюринга. Для надежности этот эксперимент может быть повторен несколько раз.

До сих пор программы, проходящей тест Тьюринга с достаточно сообразительным судьей, нет. Умелый судья может по первому же вопросу определить, кто перед ним — человек или программа.

48. Представьте, что вы — судья в тесте Тьюринга. Предложите методы ведения диалога, с помощью которых отличить человека от программы можно достаточно надежно.

Возникает важный вопросы: если машина успешно проходит тест Тьюринга, можно ли утверждать, что она ничем не отличается от человеческого ума? Достаточно ли того, что робот *ведет себя* в точности как человек (или какое-нибудь животное — неважно), для признания этого робота человеком? Нет ли у людей какого-то еще важного свойства, кроме их свойства реагировать определенным образом на внешние воздействия?

Представим себе робота (полностью алгоритмического, компьютерного — не подойдут, например, роботы с каким-нибудь уникальным hardware неалгоритмической природы), идеально похожего, ведущего себя точно так же, как и человек, и легко проходящего тест Тьюринга. Уколем этого робота иголкой. Рецепторы робота получат сигнал и начнут передавать его каким-то образом в некий обрабатывающий центр. Там этот сигнал анализируется, учитывается вся история «жизни» робота до этого момента (все это за доли секунды), и, наконец, выдается ответ: робот очень естественно и натурально изображает боль. Должны ли мы считать, что он действительно почувствовал боль? Может ли машина Тьюринга почувствовать боль? Или вообще что-нибудь почувствовать? Может ли у машины Тьюринга быть сознание?

Вероятно, для отрицательного ответа на эти вопросы не нужно прибегать к таким технически сложным рассуждениям как теорема Геделя. Приведем еще один пример. Рассмотрим клеточный автомат «жизнь» Джона Конвея. Известно, что очень простой набор правил этого автомата даже при небольшом количестве закрашенных в начале

клеток может порождать сложные ситуации и сложные последовательности состояний: паровозы, космические корабли, фильтры, фабрики, размножители, пожиратели и т.д.. Эти системы будут обладать свойствами, не наблюдающимися у отдельных клеток. Например, космические корабли будут обладать свойством «полета». Свойства системы, не заметные у ее составных частей, называются эмерджентными.

Доказано, что автомат «жизнь» алгоритмически полон, т.е. на нем можно выполнить любую программу для машины Тьюринга. Тогда сторонники возможности искусственного сознания должны согласиться, что при определенном начальном условии у наборов клеточек в «жизни» возникнет иллюзия сознания.

Уточним нашу терминологию. Свойство полета — это слабо эмерджентное свойство космического корабля: полет можно каузально свести объяснить в терминах перекраски клеток, но нельзя сказать, что полет — это и есть перекраска. Другими словами, слабо эмерджентное свойство — это свойство системы, сводящееся к ее составным компонентам каузально, но не онтологически. Примеры слабо эмерджентных свойств: температура, оптические свойства вещества, уютность дома. Все эти свойства можно каузально свести к компонентам системы: температуру объяснить через движение молекул, оптические свойства — через их форму и расположение, уютность — через взаимное расположение кирпичей.

Сильно эмерджентное свойство — это свойство системы, не сводящееся к ее составным компонентам ни онтологически, ни каузально. До сих пор никто не видел сильно эмерджентных свойств. Видимо, потому что их нет. Пример сильно эмерджентного свойства: сознание у клеточного автомата или у машины Тьюринга. В мире «жизни» нет ничего истинного или ложного. Все только *происходит*. Феномены истинности и сознания слишком фундаментальны по сравнению с феноменами, сводящимися к перекраске клеток. Можно говорить, что сознание у человека и животных — это всего лишь иллюзия. Но в мире клеточного автомата даже не может быть такой иллюзии!

6.3 Компьютерное моделирование физики

В спорах о сознании и искусственном интеллекте часто можно услышать т.н. аргумент моделирования: нет сомнений в том, что сильный ИИ возможен, т.к. можно построить компьютерную модель мозга. Конечно, аккуратное моделирование человеческого мозга на клеточном, молекулярном или еще каком-нибудь уровнях может оказаться делом неподъемным для сегодняшних технологий и даже для технологий будущего (например, если подобная модель будет требовать бит памяти больше, чем существует атомов во вселенной). Но важна принципиальная возможность промоделировать мозг.

Аргумент моделирования сталкивается со следующими проблемами:

1. Для того, чтобы построить компьютерную модель чего-то, нужно знать точные базовые законы, по которым это что-то работает. Знаем ли мы базовые физические законы, необходимые для работы мозга?

2. Компьютерная модель мозга не обязательно будет обладать всеми свойствами мозга. В частности, сознанием. Как говорит Джон Серл, «компьютерная модель зонта не защитит вас от дождя» (а защитит только от виртуального дождя). С другой стороны, компьютерная модель калькулятора считает ничуть не хуже, чем настоящий калькулятор. Сознание — это что-то вроде зонта или что-то вроде калькулятора?

3. Откуда возникла уверенность, что физические законы, даже если они нам известны, вообще могут быть промоделированы на компьютере? Разберем этот пункт более подробно.

Видимо, эта уверенность возникла вследствие успешного моделирования законов классической физики. Действительно, базовые законы классической механики, электродинамики и т.д. представляют собой дифференциальные уравнения, обыкновенные или в частных производных, для которых существуют алгоритмы приближенного решения на компьютере. С другой стороны, по любой алгоритмически неразрешимой задаче можно легко построить формальный детерминированный мир, который невозможно промоделировать никаким алгоритмом. Вопрос в том, каковы законы нашего физического мира: вычислимые или нет?

Некоторые физики, в том числе Пенроуз, предполагают, что на невычислимость природы указывает уже феномен квантовомеханического измерения. Но интерпретации квантовой механики — дело темное и спекулятивное. Оказывается, даже детерминистичная и алгоритмическая часть квантовой механики создает серьезные трудности для компьютерного моделирования. Это происходит из-за т.н. квантовой запутанности.

Рассмотрим для начала классическую систему с N частицами, эволюция которой задается какой-нибудь системой дифференциальных уравнений. Сколько памяти требуется для приближенного задания положения такой системы? Пусть для задания положения одной частицы нужно хранить в памяти n чисел. Тогда для задания положения N частиц потребуется $n + \ldots + n = nN$ чисел. Символьно это можно записать следующим образом: если V_1 — это конфигурационное пространство первой частицы, а V_2 — второй, то конфигурационным пространством системы из первой и второй частицы будет

$$V = V_1 \times V_2, \qquad \dim V_1 \times V_2 = \dim V_1 + \dim V_2$$

Сравним это с квантовой системой из N частиц. Пусть для начала $N = 1$ и для задания системы из одной частицы требуется n чисел. Пусть теперь $N = 2$. Фундаментальное свойство квантовых систем — это их способность «расплываться», т.е. находиться в линейной комбинации любых своих состояний. Другими словами, если (φ_1, φ_2) и (ψ_1, ψ_2) — состояния системы из двух частиц, то

$$a(\varphi_1, \varphi_2) + b(\psi_1, \psi_2)$$

— это тоже состояние. Таким образом, если для задания системы из одной частицы требуется, скажем, функция одной переменной, то для задания системы из двух частиц требуется уже не две функции одной переменной, как в классической механике,

а одна функция двух переменных. Поэтому память, нужная для хранения приближенного состояния системы из двух частиц — это уже не $2n$, а n^2.

Если V_1 — конфигурационное пространство первой частицы, а V_2 — второй, то конфигурационным пространством квантовой системы из первой и второй частиц будет уже не $V_1 \times V_2$, а

$$V = V_1 \otimes V_2, \qquad \dim V_1 \otimes V_2 = \dim V_1 \cdot \dim V_2$$

Это явление называется квантовой запутанностью. Из-за нее для моделирования квантовой системы из N частиц необходимо запоминать уже не nN чисел, а n^N. Квантовая запутанность делает компьютерное моделирование квантовой системы из хоть сколько-нибудь большого числа частиц (даже если предположить, что каким-то образом нам известна волновая функция этой системы в начальный момент времени) практически невозможным делом.

7 Приложения

7.1 Аксиоматика ZF теории множеств

Сейчас математики верят, что значительная часть математических доказательств может быть переведена на формальный язык исчисления предикатов с сигнатурой $(\in, =)$ и аксиомами Цермело-Френкеля. Тем не менее, лишь очень малая доля математиков способна по памяти восстановить эти аксиомы. Вот они:

1. Объемность:
$$\forall z[(z \in x \leftrightarrow z \in y) \to x = y]$$

2. Регулярность:
$$\exists y(y \in x) \to \exists y[y \in x \wedge \neg\exists z(z \in y \wedge z \in x)]$$

3. Неупорядоченная пара:
$$\exists z[\forall u(u \in z \leftrightarrow u = x \vee u = y)]$$

4. Объединение:
$$\exists z[\forall u(u \in z \leftrightarrow \exists y(y \in x \wedge u \in y))]$$

5. Множество всех подмножеств:
$$\exists z[\forall y(y \in z \leftrightarrow y \subset x)]$$

6. Схема выделения:
$$\exists u \forall x[x \in u \leftrightarrow (x \in y \wedge A(x, \overline{w}))]$$

7. Схема замены:

$$\forall x \in u \exists! y A(x, y, \overline{w}) \to \exists z \forall y (y \in z \leftrightarrow \exists x \in u A(x, y, \overline{w}))$$

8. Бесконечность:

$$\exists y [\emptyset \in y \wedge \forall x (x \in y \to x \cup \{x\} \in y)]$$

Для краткости в записи аксиом мы допустили некоторые вольности: скобки поставлены не везде и для всех свободных переменных нужно добавить перед аксиомами кванторы всеобщности. Также использованы сокращения: $\exists!$, $\subset$, $\cup$, $\emptyset$, $\{x\}$. Например, $y \subset x$ нужно заменить на $\forall u (u \in y \to u \in x)$ Некоторые из этих аксиом можно отбросить (т.к. схема аксиом замены довольно сильная).

49. Перескажите аксиомы ZF на русском языке.

50. В 1963 году математик Пол Коэн, опираясь на работы Геделя, доказал, что ни континуум-гипотеза, ни ее отрицание не могут быть выведены в системе ZF. Может ли подобное утверждение быть доказано и про гипотезу Гольдбаха? Другими словами, возможно ли доказательство (скажем, в ZF) того, что ни сама гипотеза Гольдбаха, ни ее отрицание не доказуемы в PA?

7.2 Аксиоматика Тарского евклидовой геометрии

Несмотря на то, что аксиоматика Евклида была, по-видимому, первой математической (неформальной) системой аксиом, аксиоматизация геометрии на формальном уровне (когда новые утверждения получаются лишь с помощью синтаксических операций из предыдущих), оказалась непростым делом. В 1899 году формальную систему аксиом евклидовой геометрии придумал Гильберт, но его аксиомы нельзя было сформулировать на языке первого порядка. Когда была создана аксиоматика ZF (1908-1922), появилась возможность определить действительные числа и плоскость $\mathbb{R}^2$ как определенного вида множества. Построить аксиоматику геометрии, не зависящую от более общих теорий типа теории множеств и оставаясь в рамках логики первого порядка, удалось лишь в 1959 году Альфреду Тарскому.

В отличие от Гильберта, который в качестве примитивных объектов использовал точки и прямые, аксиоматика Тарского подразумевает лишь один тип примитивных объектов: точки. Сигнатура же языка состоит из трехместного предикатного символа B и четырехместного предикатного символа $\equiv$. Стандартная модель системы аксиом Тарского такова: носителем является евклидова плоскость $\mathbb{R}^2$, $B(x, y, z)$ означает, что точка y лежит на отрезке xz, а $xy \equiv zt$ означает, что расстояние между точками x и y равно расстоянию между z и t. Аксиомы:

1. $xy \equiv yx$
2. $xy \equiv zz \to x = y$
3. $(zu \equiv xy \wedge yx \equiv wv) \to zu \equiv wv$
4. $Bxyx \to x = y$

5. На луче можно отложить отрезок данной длины:

$$\exists a(Bwxa \wedge xa \equiv yz)$$

6. Аксиома Паша:

$$(Bxuz \wedge Byvz) \rightarrow \exists a(Buay \wedge Bvax)$$

7. Ограничение размерности снизу (существуют три неколлинеарные точки):

$$\exists a \exists b \exists c(\neg Babc \wedge \neg Bbca \wedge \neg Bcab)$$

8. Ограничение размерности сверху (срединный перпендикуляр к двум точкам является прямой):

$$(xu \equiv yv \wedge yu \equiv yv \wedge zu \equiv zv \wedge u \neq v) \rightarrow (Bxyz \vee Byzx \vee Bzxy)$$

9. Пятый сегмент:

$$(x \neq y \wedge Bxyz \wedge Bx'y'z' \wedge xy \equiv x'y' \wedge yz \equiv y'z' \wedge xu \equiv x'u' \wedge yu \equiv y'u') \rightarrow zu \equiv z'u'$$

10. Пятый постулат Евклида:

$$(Bxuv \wedge Byuz \wedge x \neq u) \rightarrow \exists a \exists b(Bxya \wedge Bxzb \wedge Bavb)$$

11. Схема непрерывности:

$$\exists a \forall x \forall y[(\varphi(x) \wedge \psi(y)) \rightarrow Baxy] \rightarrow \exists b \forall x \forall y[(\varphi(x) \wedge \psi(y)) \rightarrow Bxby]$$

Тарский доказал, что все модели этой системы аксиом изоморфны между собой. Следовательно, по теореме Геделя о полноте, система аксиом Тарского полна. Тарский также установил, что его теория разрешима: существует алгоритм, для любого замкнутого синтаксически правильного предложения сигнатуры $(B, \equiv)$ определяющий его истинность в стандартной интерпретации. Для построения такого алгоритма было достаточно научиться переводить любое предложение сигнатуры $(B, \equiv)$ на язык полиномиальных уравнений и неравенств, где уже можно применить алгоритм Зайденберга-Тарского.

7.3 Аннотация для ЛМШ

Первая теорема Геделя о неполноте является, пожалуй, самой известной теоремой математики XX века. Она утверждает, что при любом определении формального доказательства найдутся истинные, но не доказуемые утверждения. Тем самым математику невозможно свести к синтаксической игре. Согласно второй теореме о неполноте, непротиворечивость достаточно общих систем аксиом, типа системы аксиом ZF теории множеств, невозможно доказать финитными методами. На курсе мы познакомимся с

основными понятиями и феноменами математической логики и теории алгоритмов (языки первого порядка, алгоритмически неразрешимые задачи...) и, используя этот аппарат, строго докажем теоремы о неполноте. Мы увидим, как, с одной стороны, конкретные математические теоремы могут иметь большое значение для философии, а с другой, какую роль в математике играют философские идеи (логический позитивизм, категориальная система Канта, нотация Рассела, проблема бороды Платона и гипотеза сильного ИИ).

Список литературы

[1] Н. К. Верещагин, А. Шень. Языки и исчисления.

[2] Н. К. Верещагин, А. Шень. Вычислимые функции.

[3] R. Feynman. Simulating physics with computers.

[4] T. Franzen. Godel's theorem — an incomplete guide to its use and abuse.

[5] Ю. И. Манин. Теорема Геделя.

[6] R. Penrose. The Emperor's New Mind.

[7] Платон. Парменид.

[8] W. V. Quine. On what there is.

[9] А. Л. Семенов. Математика текстов.

[10] J. Searle. The Rediscovery of the Mind.

[11] Б. А. Трахтенброт. Алгоритмы и машинное решение задач.

[12] A. M. Turing. Computing machinery and intelligence.

[13] В. А. Успенский. Теорема Геделя о неполноте и четыре дороги, ведущие к ней.