Введение в математическую логику

Лекция 3

1. Универсальный алгоритм и его применение для решения поставленных в прошлых лекциях проблем

1.1. Универсальный алгоритм Маркова

Напомню, что мы оперируем, с одной стороны, понятием алгоритма, не определяемым через другие математические понятия, с другой стороны, мы используем специальный вид алгоритмов — алгоритмы Маркова, определяемые формально математически. Алгоритмы Маркова по своему описанию напоминают грамматики. Однако всякий алгоритм Маркова, как и вообще всякий алгоритм, задает (детерминированный) процесс. Процесс этот может не заканчиваться, но заканчиваясь, дает результат, однозначно определенный исходным данным.

Мы сформулировали и приняли тезис Чёрча, в соответствии с которым всякая функция, преобразующая слова в слова, вычисляемая некоторым алгоритмом, может быть вычислена алгоритмом Маркова.

Следующим важным понятием, с которого мы сегодня начнем, является понятие универсального алгоритма.

Дадим сначала некоторые содержательные пояснения.

Когда я вам объяснял, что такое алгоритм Маркова, то при этом исходил из того, что вы можете воспроизвести, в принципе, работу любого такого алгоритма. Сейчас мы свяжем с этим некоторые формальные определения.

Начнём с того, какие в алгоритме Маркова использовались символы. Это были символы 0, 1, — основной алфавит, и, кроме того, в определении алгоритма использовались ещё стрелочка и точка, и дополнительные символы, различные для различных алгоритмов (но для любого заданного алгоритма их алфавит фиксирован).

Вспомним еще, что мы говорили о возможности кодирования самых разных объектов словами, например, в алфавите $\{0, 1\}$. В частности, такими словами можно закодировать и всякую программу алгоритма Маркова.

Упорядочим алфавит алгоритмов Маркова:

$$0, \quad 1, \quad \longrightarrow, \quad \bullet, \quad A_1, \quad A_2, \quad \ldots \quad .$$

Закодируем все правила алгоритма в двоичном алфавите. Программа алгоритма — двоичный код набора всех правил (или соответствующее число).

Представим себе, что вы сами получаете код какого-то алгоритма Маркова и исходное данное для этого алгоритма. Тогда Вы можете, чисто механически начать применять этот алгоритм к этому исходному данному — слову, и продолжать этот процесс до получения того результата, который должен получить данный вам алгоритм, или до бесконечности, если этот алгоритм не кончает работу.

Мы сейчас описали некоторый алгоритм. Уточним, что является его исходным данным. Исходя из нашего описания, исходное данное — это пара слов, скажем Π (программа) и x (аргумент) в алфавите $\{0, 1\}$. Но пары слов, как мы помним, можно тоже кодировать словами в алфавите $\{0, 1\}$. Наш алгоритм задает

вычислимую функцию. Она называется универсальной функцией для алгоритмов Маркова. Универсальная функция U для всякого алгоритма Ф с программой П и слова x дает результат $U(\text{П}, x)$ применения Ф к x.

В силу тезиса Чёрча, для этого алгоритма существует алгоритм Маркова, вычисляющий ту же функцию.

Андрей Андреевич Марков, придумавший алгоритмы Маркова (он называл их: «нормальные алгорифмы») тщательно выписал в явном виде построение алгоритма, вычисляющего универсальную функцию, и доказал правильность его работы. При этом он открыл две важные для практического программирования вещи. Первое его открытие состояло в указании основных конструкций структурного программирования — последовательного выполнения, разветвления и итерации. (К ним он еще добавлял, вполне разумно, и операцию «заведения новой ячейки памяти».)

Второе его открытие состояло в изобретении методов доказательства правильности программы (сейчас это называется индукцией с использованием инвариантов). Марков доказал правильность работы своего алгоритма, вычисляющего универсальную функцию. Сегодня мы назвали бы это правильностью работы компилятора. Андрей Андреевич Марков был тщательный и дотошный человек, это его доказательство заняло несколько сотен страниц.

Среди других его достижений было и построение некоторой части математики, в частности, математического анализа, исходя из представлений о том, что не следует рассматривать просто существование какого-нибудь математического объекта, а нужно иметь в виду какой-то способ его построения, конструирования. Такой подход к математике называется конструктивизмом, получаемая математика — конструктивной. Получается, например, конструктивный математический анализ, имеющий дело с числами, которые задаются алгоритмически, и с функциями, которые тоже алгоритмически задаются.

1.2. Канторова диагональ

В прошлый раз мы сформулировали некоторые проблемы, и сегодня мы выясним, как они решаются.

Начну с очень простого примера. Пусть у нас есть таблица, например, размером 5 на 5, и ее строками являются последовательности.

Номер члена последовательности / Номер последовательности	0	1	2	3	4
0	**1**	2	8	2	3
1	0	**3**	7	3	5
2	7	6	**0**	3	9
3	8	2	7	**5**	8
4	3	0	6	5	**0**

В нулевой строке, например, мы видим последовательность 1, 2, 8, 2, 3, и т. д. Как построить последовательность, которой нет в этой таблице?

Можно это делать по-разному. Можно просмотреть всю таблицу, выяснить, например, что там все числа меньше ста, и написать строку из одних сотен.

Можно, если нам трудно сделать глобальное утверждение обо всей таблице, идти по диагонали и строить последовательность чисел, члены которой получаются изменением чисел, стоящих на диагонали. Например, можно ко всем числам на диагонали добавлять по 1. Если мы к рассмотренной нами в качестве примера таблице применим этот способ, добавляя ко всем элементам на диагонали 1, то получим последовательность 2, 4, 1, 6, 1. Этой последовательности нет в таблице просто потому, что она от каждой строчки отличается в диагональном элементе.

Это тот же самый приём, который используется в доказательстве несчётности множества последовательностей из нулей и единиц и несчётности множества действительных чисел. Прием этот изобрел немецкий математик Георг Кантор. Он показал, что невозможна (уже бесконечная) таблица, строки которой пронумерованы натуральными числами и среди строк которой встречаются все последовательности нулей и единиц.

Номер цифры / Номер последовательности	0	1	2	3	4	…
0	**1**	0	0	1	0	…
1	1	**1**	0	1	0	…
2	1	0	**0**	1	1	…
3	0	0	0	**1**	0	…
4	0	1	0	1	**0**	…
…	…	…	…	…	…	…

Действительно, для всякой такой таблицы можно рассмотреть последовательность, стоящую на ее диагонали и изменить каждый её член, заменив 0 на 1, а 1 на 0. В нашем примере диагональная последовательность:

$$1, 1, 0, 1, 0, \ldots,$$

а соответствующая ей изменённая последовательность:

$$0, 0, 1, 0, 1, \ldots.$$

Про саму диагональную последовательность мы не знаем, есть ли она в бесконечной таблице, но измененной диагональной последовательности в таблице уже точно не будет: она от первой строки отличается на первом месте, от второй строки отличается на втором месте, от третьей строки отличается на третьем месте и т. д.

Вот это знаменитое доказательство несчётности множества действительных чисел или последовательностей нулей и единиц («Канторову диагональ») мы и используем в теории алгоритмов.

1.3. Построение невычислимой функции. Функция вычислимая, но не продолжаемая до всюду определённой вычислимой функции. Перечислимое неразрешимое множество

Рассмотрим бесконечную таблицу универсальной функции $U(\Pi, x)$. Строки в ней соответствуют последовательно выписанным двоичным словам, которые являются кодами программ. Двоичные слова, которыми помечены столбцы таблицы, являются входными данными для программ. В клетках таблицы записаны результаты применения программ к соответствующим входным данным. Некоторые клетки пустые, потому что функция в соответствующей строке не определена для аргумента из соответствующего столбца.

Аргумент / Программа	Λ	0	1	00	01	…
Λ	01	0		010		…
0		1		0100		…
1	101		000		000	…
00		0			101	…
01		1			0	…
…	…	…	…	…	…	…

Пример. Так приблизительно может выглядеть бесконечная таблица универсальной функции

Поскольку функция универсальна, то в этой таблице в качестве строчек имеются все вычислимые функции. Построим теперь диагональную функцию, а именно применим функцию U к одному и тому же аргументу в качестве программы и в качестве исходного данного. Выглядит это немножко экзотически, но ничего страшного в этом нет. И то и другое — двоичные последовательности, и мы вычисляем $U(n, n)$. Допишем ко всем значениям этой функции единицу в конце. Это — новая — «диагональная» функция. Будет ли она вычислимой? Конечно, будет. Ведь универсальная функция вычислима, мы вполне можем подставить в неё два одинаковых аргумента, и потом можем добавить единицу. Ничего в этом удивительного нет. Но будет ли она находиться в этой таблице? Понятное дело, будет. В этой таблице имеются в качестве строчек все вычислимые функции. Получим ли мы при этом противоречие, похожее на то, которое мы только что встретили, когда обсуждали несчётность последовательностей нулей и единиц? Действительно, в той самой строчке, где написана эта функция, она будет отличаться на диагональной клетке от себя самой. Верно это или нет?

Конечно же, в этой диагональной клетке, в строчке, где выписана построенная нами функция, не написано ничего. Если к этому «ничему» приписать 1, то будет

снова «ничего» — «диагональная» функция не определена. Никакого парадокса в том, что наша функция имеется в этой таблице, нет.

Что будет, если в диагональной клетке «диагональной» функции что-то написать, например, 0? Будет ли получившаяся функция вычислимой? Конечно, будет (это очевидно). Однако она будет стоять уже в другой строчке таблицы.

Рассмотрим какую-то функцию, которая получается из $U(n,n)$ продолжением на все слова в алфавите $\{0, 1\}$. Например, можно доопределить ее всюду нулем. Будет ли эта функция вычислимой? Предположим, что — да. Проделаем с ней ту же трансформацию, допишем ко всем значениям 1. Полученная функция тоже должна быть вычислимой, но не встречается в таблице в силу того рассуждения, которое мы уже проделывали. Значит, никакое доопределение функции $U(n,n)$ на все слова не является вычислимой функцией.

Итак, мы построили пример невычислимой функции, которую нам удалось «явно описать», начиная, скажем, с универсальной функции U для алгоритмов Маркова.

Рассмотрим теперь множество слов n, для которых $U(n,n)$ определена. Это множество — перечислимо. Мы это выясняли на прошлой лекции. А что можно сказать о множестве тех слов, для которых $U(n,n)$ НЕ определена? Если бы оно тоже было перечислимым, то оно (и его дополнение) было бы разрешимым, как мы доказывали на прошлой лекции. Но тогда можно было бы продолжить $U(n,n)$ на все слова, сначала проверяя определенность, а потом или вычисляя $U(n,n)$, или беря в качестве значения (например) 0.

Итак, мы построили пример вычислимой функции, не продолжаемой до всюду определенной вычислимой функции, и пример перечислимого множества с неперечислимым дополнением, то есть, множества — перечислимого и неразрешимого. Это и отвечает на поставленные в прошлый раз вопросы.

На этом наше введение в теорию алгоритмов — теорию вычислимых функций, разрешимых и перечислимых (породимых) множеств заканчивается. Но с диагональной конструкцией мы еще встретимся в теореме Гёделя и теории сложности.

2. Логика высказываний

2.1. Синтаксис логики высказываний. Индуктивное определение формулы

Логика высказываний — это базовая часть логики, логическая система, которая встречается в самых разных ситуациях, в более сложных логиках, является основной для этих более сложных и нетривиальных логик, и мы, естественно, начнём с неё.

Логика занимается истинностью и ложностью утверждений, их доказуемостью и недоказуемостью.

Начнём с того, что у нас есть имена для истины и лжи, или логические имена, — это ноль и единица. При этом 0 будет соответствовать в нашей модели мира, того, что мы моделируем с помощью математики, лжи. Иногда пишется просто слово «Ложь», иногда буква «Л», иногда английское «F» от слова «False». В качестве имени истины используется 1, соответственно «Истина», «И», или «T» от слова «Truth». Нам удобно использовать 0 и 1, в частности, потому, что это элементы нашего базового алфавита.

Ещё нам понадобится упорядоченный (бесконечный) алфавит логических переменных: $A_0,\ A_1,\ A_2, \ldots$. Помните, мы работали с термами, и там тоже были переменные для объектов, были имена для функций, а здесь нам понадобятся логические переменные. Смысл логических переменных — мы к семантике вернёмся позже — состоит в том, что значениями этих переменных являются ноль или единица.

Ещё нам понадобятся логические связки. Пока это просто символы, значки с названиями. Потом мы поговорим о том, как из этих значков строить формулы, и это называется синтаксисом, как вы помните. Ещё поговорим о смысле этих формул, об их интерпретации, и это называется семантикой.

Логические связки:

$\neg$ — отрицание, «не» (иногда логики называют эту связку «кочергой»),
$\wedge$ — конъюнкция, «и»,
$\vee$ — дизъюнкция, «или»,
$\longrightarrow$ — импликация, «влечет», «если. . . , то. . . »,
$\equiv$ — эквивалентность, «равносильно».

Теперь дадим индуктивное определение формулы. (Вы помните, как мы индуктивно определяли терм.)

Индуктивное определение формулы:

- Логические имена, логические переменные — формулы.
- Если $\Phi,\ \Psi$ — формулы, $\tau \in \{\wedge, \vee, \longrightarrow, \equiv\}$, то $(\neg\Phi),\ (\Phi\tau\Psi)$ — формулы.

Как и в случае обычных арифметических формул, для логических формул можно ввести правила их сокращенной записи. Если использовать эти правила, то формулы становятся короче, их легче писать и читать. Конечно, нужно следить за тем, чтобы по сокращенной записи можно было восстановить исходную — до сокращения. Мы договоримся об опускании некоторых скобок в формулах:

- Будем опускать самые внешние скобки.
- Формулу $(\neg A_i)$ будем записывать $\neg A_i$.
- Аналогично тому, как в арифметике $a + b \cdot c$ расшифровывается как $a + (b \cdot c)$,
  в логике $a \vee b \wedge c$ расшифровывается, как $a \vee (b \wedge c)$,
  формула $a \longrightarrow b \vee c$ расшифровывается, как $a \longrightarrow (b \vee c)$,
  формула $a \longrightarrow b \equiv c$ — как $(a \longrightarrow b) \equiv c$,
  а про формулу $a \wedge b \wedge c$ можно считать, что она расшифровывается,
  например, как $a \wedge (b \wedge c)$.

Мы не будем активно использовать сложные логические формулы, и нам не понадобятся детальные правила опускания скобок.

Задача (для самостоятельного решения). Выписать правило опускания скобок и построить алгоритм, который восстанавливает в формуле опущенные скобки.

2.2. Семантика формул логики высказываний

Перейдем к семантике формул. При содержательном понимании логические переменные означают высказывания, а логическим связкам отвечают союзы и другие конструкции естественного языка в соответствии с названиями, приведенными выше. Однако в математике мы определяем семантику формул, интерпретируя связки, как операции над элементами 0 и 1.

Посмотрим на таблицу:

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \longrightarrow B$	$A \equiv B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Как она устроена? В первых двух столбцах имеются все четыре возможные комбинации значений A и B.

В третьем столбце мы видим значения для $\neg A$: эти значения противоположны значению A. В следующем столбце мы видим, как вычисляется значение конъюнкции. Мы видим, что если A и B оба равны 0, то получается 0, и более того, если хотя бы одно только A равно 0, или только B равно 0, то конъюнкция равна 0. Это соответствует нашему интуитивному представлению о том, что значит, что «A и B» истинно: «A и B» истинно, когда и A истинно, и B истинно, то есть обе переменные одновременно истинны.

Следующая операция — дизъюнкция, «или». «A или B» истинно, если или A, или B — истинно. Дизъюнкция оказывается ложной только когда оба аргумента этой операции, и A, и B — ложны.

Импликация. Здесь немножко сложнее. В какой ситуации «Если A, то B» истинно? Из таблицы видно, что если A — ложно, то импликация истинна. Значит, если «единица равняется нулю», то «три в квадрате равняется 16». Верно это или нет? Можно долго пытаться понять, как наиболее разумно подойти к этому вопросу, но математики подошли к нему очень просто. Они договорились, что значок, который выглядит как стрелочка, означает операцию, которая дает 0, только если первый аргумент 1, а второй 0. Первый аргумент импликации называют *посылкой* импликации, а второй — *заключением.* Итак, если мы обнаружили ложность посылки, то вся импликация истинна.

И последний столбец таблицы относится к эквивалентности. Как и следовало ожидать от эквивалентности, «A эквивалентно B» истинно тогда и только тогда, когда A и B имеют одинаковые значения.

Следующим шагом должно быть определение семантики формул, то есть как, пользуясь операциями определить, что такое смысл, семантика, любой формулы.

Фиксируем бесконечную последовательность

$$\alpha = \alpha_0, \alpha_1, \alpha_2, \ldots, \qquad \text{где } \alpha_i \in B = \{0, 1\}.$$

Значение формулы на данной последовательности определяем индукцией по построению формулы:

1. Значением логического имени является оно само.

2. Значением логической переменной A_i является α_i.

3. Значением формулы $(\Phi \tau \Psi)$, где $\tau \in \{\wedge, \vee, \longrightarrow, \equiv\}$, является результат применения операции τ к значениям формул Φ, Ψ. Значением формулы $(\neg\Phi)$ является отрицание значения формулы Φ, т. е. $\mathbf{Зн}(\neg\Phi) = 1 - \mathbf{Зн}\,\Phi$.

Зададимся следующим вопросом: однозначно ли определено значение формулы на данной последовательности? Ответ не совсем очевиден и вытекает из «однозначности анализа» формулы. Более подробно, нужно доказать, что любая формула может быть единственным образом представлена или в виде $(\Phi\tau\Psi)$, или в виде $(\neg\Phi)$. Единственность представления вытекает из следующих двух утверждений, доказываемых индукцией по построению формулы:

1. Всякая формула начинается со скобки. Отсюда вытекает невозможность одновременного представления в первом и втором виде.

2. Во всякой формуле число открывающих скобок «(» равно числу закрывающих скобок «)», во всяком начале формулы, с ней не совпадающим, число открывающих скобок больше числа закрывающих. Отсюда вытекает, что среди начальных отрезков (фрагментов) формулы $(\Phi\tau\Psi)$ мы только для фрагментов «$(\Phi$», «$(\Phi\tau$» и «$(\Phi\tau\Psi$» получим разность между числом открывающих и закрывающих скобок, равную 1. Поскольку всякая формула кончается скобкой, то для выделения Φ (а, значит, и τ, Ψ) есть единственная возможность.

Если не фиксировать последовательность α, а рассматривать разные последовательности, то получим, что значением формулы является функция $B^\omega \longrightarrow B$, которую мы называем значением формулы. Пусть наибольший номер (индекс) переменной в формуле равен $n-1$. Тогда формула задает также функцию $B^n \longrightarrow B$.

Математическая логика была исходно, я об этом говорил на первой лекции, ориентирована, прежде всего, на анализ математических утверждений, математических рассуждений, алгоритмов и т. д. Но конечно, логика, как тоже уже упоминалось, — это наука куда более старая и почтенная, чем математическая логика. В частности, знаменитый философ Аристотель сформулировал некоторое количество логических конструкций, некоторые из них мы, может быть, ещё упомянем в нашем курсе. Конструкции, т. н. «силлогизмы» Аристотеля, учили студенты средневековых университетов, да и позднее тоже, в курсе логики. Один из знаменитых примеров силлогизма, в котором вместо переменных уже подставлены конкретные высказывания:

«Все люди смертны. Сократ — человек. Следовательно, Сократ смертен.»

В приведенном примере силлогизма, как и в других, существенную роль играет слово «все». Анализом формул, включающих символы, соответствующие словам «все», «существует», мы будем заниматься в дальнейшем.

Когда мы начинаем анализировать тексты естественного языка, отдельные предложения, то может возникнуть противоречие между математическим смыслом и общепринятым. Одним из примеров этого является ситуация с «или». Когда мы говорим: «Ты пойдёшь в кино или на хоккей?», мы не имеем в виду, что может случиться и то, и другое. И ответ «и туда, и туда» будет для нас выглядеть невозможным, то есть утверждение, что я пойду и в кино, и на хоккей, будет, скорее всего, рассматриваться как ложное. В естественном языке часто и союз «и», и союз «или», и импликация понимаются иначе, чем в математике. Относительно утверждений типа: «если Луна сделана из зелёного сыра, то я испанский король», можно долго выяснять, истинно оно, или ложно, или бессмысленно. Часто в импликации предполагается какая-то связь между посылкой и заключением. Вот ничего этого не имеется в виду в математического логике, в ней говорится просто о формальных таблицах.

Предположим, что у нас имеется три логических переменных A_0, A_1, A_2 и мы рассматриваем различные комбинации их значений. Таких комбинаций восемь, 2^3. Пусть теперь имеется формула, например,

$$(A_0 \longrightarrow \neg A_1) \wedge \neg(A_2 \vee A_0).$$

Чтобы найти соответствующую функцию, надо вычислить значение этой формулы на каждой из указанных комбинаций. Вычислять надо индукцией по построению, начиная с атомных формул.

A_0	A_1	A_2	$(A_0$	$\longrightarrow$	$\neg$	$A_1)$	$\wedge$	$\neg$	$(A_2$	$\vee$	$A_0)$
0	0	0		1	1		**1**	1		0	
0	0	1		1	1		**0**	0		1	
0	1	0		1	0		**1**	1		0	
0	1	1		1	0		**0**	0		1	
1	0	0		1	1		**0**	0		1	
1	0	1		1	1		**0**	0		1	
1	1	0		0	0		**0**	0		1	
1	1	1		0	0		**0**	0		1	

Таблица функции. Построение функции по формуле. Под каждой логической связкой написано значение соответствующей подформулы. Под главной связкой (в данной формуле — это конъюнкция) — значение самой формулы (жирным шрифтом).

Наша формула — это конъюнкция двух формул. Строим таблицу функции для каждой из них и т. д.

2.3. Дизъюнктивная нормальная форма

Введем еще некоторые обозначения. Для обозначения дизъюнкции или конъюнкции многих членов используются те же самые способы записи, которые используются для длинных сумм $\sum a_i$ или произведений $\prod a_i$. Пишется знак дизъюнкции или конъюнкции, а после этого некоторым образом задается список формул, между которыми стоит этот знак:

$$\bigvee \Phi_i, \qquad \bigwedge \Phi_i.$$

Заметим, что операции конъюнкции и дизъюнкции, как и сложения, и умножения, являются ассоциативными и коммутативными.

Теперь посмотрим на некоторые специального вида формулы. Начнем с конъюнкций логических переменных и их отрицаний. Соответствующая функция принимает значение 1 только для одной комбинации переменных (в одной точке). Это очевидно. Например, где функция, задаваемая формулой $A_0 \wedge \neg A_1 \wedge \neg A_2 \wedge A_3$ равна 1? — Только при значении $\langle \alpha_0, \alpha_1, \alpha_2, \alpha_3 \rangle = \langle 1, 0, 0, 1 \rangle$.

Фиксируем натуральное число n. Обозначим

$$\alpha = \alpha_0, \alpha_1, \ldots, \alpha_{n-1},$$

$$A = A_0, A_1, \ldots, A_{n-1}.$$

Далее, введем обозначения:

$$(0, A_i) = \neg A_i,$$

$$(1, A_i) = A_i,$$

$$(\alpha, A) = ((\alpha_0, A_0) \wedge (\alpha_1, A_1) \wedge \ldots \wedge (\alpha_{n-1}, A_{n-1})).$$

Теорема о дизъюнктивной нормальной форме.
Всякая функция $f : B^n \longrightarrow B$ задается формулой

$$\bigvee(\alpha, A), \quad \text{по всем } \alpha, \text{ для которых } f(\alpha) = 1.$$

Всякая формула указанного вида называется дизъюнктивной нормальной формой — Д.Н.Ф. Если в каждой конъюнкции, входящей в рассматриваемую дизъюнкцию, присутствуют все переменные A_0, A_1, $\ldots$, A_{n-1}, то формула называется совершенной дизъюнктивной нормальной формой (С.Д.Н.Ф.).