Блатов И.А., Старожилова О.В.

Математическая логика и теория алгоритмов

Самара
2017

ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ
Федеральное государственное образовательное учреждение высшего образования

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра высшей математики

Блатов И.А., Старожилова О.В.

Математическая логика и теория алгоритмов

Учебное пособие

Самара,

2017

Содержание

Введение

«математическая логика есть логика по предмету, математика по методу».

Б. Порецкий

Своим существованием наука «алгебра логики» обязана английскому математику Джорджу Булю, который исследовал логику высказываний. Первый в России курс по алгебре логики был прочитан Б. Порецким в Казанском государственном университете.

Как самостоятельная наука логика оформилась в трудах греческого философа Аристотеля (384-322 г.г до н.э.). Он систематизировал известные до него сведения, и эта система стала впоследствии называться формальной или Аристотелевой логикой.

Формальная логика просуществовала без серьёзных изменений более двадцати столетий. Естественно, что развитие математики выявило недостаточность Аристотелевой логики и потребовало дальнейшего её развития.

Впервые в истории идеи о построении логики на математической основе были высказаны немецким математиком Г. Лейбницем (1646-1716) в конце XVII века. Он считал, что основные понятия логики должны быть обозначены символами, которые соединяются по особым правилам. Это позволит всякое рассуждение заменить вычислением.

"Мы употребляем знаки не только для того, чтобы передать наши мысли другим лицам, но и для того, чтобы облегчить сам процесс нашего мышления" (Лейбниц).

Первая реализация идеи Лейбница принадлежит английскому учёному Д. Булю (1815-1864). Он создал алгебру, в которой буквами обозначены высказывания, и это привело к алгебре высказываний. Введение символических обозначений в логику имело для этой науки такое же решающее значение, как и введение буквенных обозначений для математики. Именно благодаря введению символов в логику была получена основа для создания новой науки – математической логики.

Применение математики к логике позволило представить логические теории в новой удобной форме и применить вычислительный аппарат к решению задач, малодоступных человеческому мышлению, и это, конечно, расширило область логических исследований.

Одной из основных причин развития математической логики является широкое распространение аксиоматического метода в построении различных математических теорий.

Большинство интерпретаций для математических теорий (и, в частности, для арифметики) строится на базе теории множеств.

Математическая теория, непротиворечивость которой требовалось доказать, стала предметом другой математической теории, которую Гилберт назвал *метаматематикой*, или *теорией доказательств*.

Выбирая по-разному системы аксиом и правила вывода одних формул из других, получают различные синтаксические логические теории. Каждую из них называют логическим исчислением.

В последнее время большое внимание уделяется изучению сложности алгорифмов. Так, например, недавно было показано, что арифметика сложения натуральных чисел, являющаяся разрешимой теорией, может иметь только очень сложные разрешающие алгорифмы.

На практике множество элементарных логических операций является обязательной частью набора инструкций всех современных микропроцессоров и соответственно входит в языки программирования. Это является одним из важнейших практических приложений методов математической логики.

Задача изучения дисциплины – формирование логического мышления, развитие абстрактного мышления, освоение аппарата математической логики.

Глава 1 Классическая логика

Современная логика развилась в точную науку, применяющую математические методы.

Предметом исследования в логике являются человеческие рассуждения, а основная задача ее состоит в выявлении правильных способов рассуждений. При этом под рассуждением понимается переход от некоторых суждений, относящихся к определенному предмету или вопросу и называемых посылками, к суждению, называемому заключением и полученному таким путем, что любое текущее суждение является следствием предшествующих суждений.

Классическая логика — термин, используемый в математической логике по отношению к той или иной логической системе, для указания того, что для данной логики справедливы все законы (классического) исчисления высказываний.

Неклассическая логика соответственно есть логика, в которой один или несколько законов классической логики не выполняются.

Нередко приставку **классическая** употребляют также по отношению к некоторым неклассическим логикам, которые допускают несколько вариантов — с законом исключения третьего (или подобных ему) и без. Тогда первую называют классической.

Перед изучением основ математической логики необходимо рассмотреть понятие логики в целом.

Logos (греч.) - слово, понятие, рассуждение, разум.

Слово "логика" обозначает совокупность правил, которым подчиняется процесс мышления или обозначает науку о правилах рассуждения и тех формах, в которых оно осуществляется.

Логика изучает абстрактное мышление как средство познания объективного мира, исследует формы и законы, в которых происходит отражение мира в процессе мышления.

Этапы развития логики

1-й этап связан с работами ученого и философа **Аристотеля** (384-322 гг. до н.э.). Он пытался найти ответ на вопрос "как мы рассуждаем", изучал "правила мышления".

Аристотель впервые дал систематическое изложение логики. Он подверг анализу человеческое мышление, его формы - понятие, суждение, умозаключение и рассмотрел мышление со стороны строения, структуры, то есть с формальной стороны. Так возникла **формальная логика** или **Аристотелева логика.**

➢ **Определение Формальная логика** - направление в математической логике связанное с анализом обычных содержательных умозаключений, выражаемых разговорным языком.

Учение о силлогизме впервые изложено у Аристотеля в его «Первой аналитике», где он исследовал различные формы рассуждений и их комбинаций, ввел понятие **силлогизма,** т.е. рассуждения, в котором из заданных двух суждений выводится третье. Силлогизм (от греч. sillogismos) категорический.

➢ **Определение Классическая логика** – логика, основанная на силлогизмах.

В силлогистике Аристотеля посылки и заключения формируются в виде стандартных категорических утверждений, например, "все S суть P ", "некоторые S не суть P " и др. Чтобы доказать правильность рассуждения средствами теории Аристотеля, необходимо было все суждения предоставить в форме простых категорических высказываний.

Теория правильных рассуждений, которая называлась **силлогистикой** и метод вывода правильных заключений из посылок – **дедукция** были сформированы более 2 тысяч лет тому назад, никем не опровергнуты и считаются присущими человеческому мышлению.

✓ **Замечание** Не путать с "дедуктивным методом" Шерлока Холмса. У Холмса, или скорее у Конан-Дойля, явно были проблемы с логикой, коль скоро он путал дедукцию с индукцией...

ДЕДУКТИВНЫЙ подход, называемый еще АКСИОМАТИЧЕСКИМ, это подход от общего к частному. От аксиом (постулатов) к теоремам (следствиям).

- **Пример** силлогизм правильной формы:

Все квадраты – ромбы → все ромбы - параллелограммы→Следовательно, все квадраты - параллелограммы.

В общем виде этот силлогизм имеет форму: *"Все **а** суть **в**, все **в** суть **с**. Следовательно, все **а** суть **с**."*

- **Пример** силлогизма неправильной формы:

Все квадраты - ромбы. →Некоторые ромбы имеют острый угол. →Следовательно, некоторые квадраты имеют острый угол.

- Значит, силлогизм, имеющий форму *"Все **а** суть **в**, некоторые **в** суть **с**. Значит, некоторые **а** суть **с**"* может привести к ложным выводам.

- **Определение Модус** - разновидность некоторой общей схемы рассуждения, разновидность силлогизмов.

Силлогизмы делятся по логической форме посылок и заключения — на *модусы*.

- **Пример** правильный модус:

Все фрукты питательны.

Все фрукты вкусны.

Некоторые вкусные продукты питательны

Аристотель выделил все правильные формы силлогизмов, которые можно составить из рассуждений вида:

1. **"Все а суть в"**
2. **"Некоторые а суть в"**
3. **- "Все а не суть в"**
4. **"Некоторые а не суть в"**

Таким образом, имеются правильные и неправильные силлогизмы (*модусы*). Из 256 возможных силлогизмов только 24 являются правильными, а остальные могут привести к ошибочному выводу.

Правильные модусы образуют ядро теории дедуктивных выводов, в котором от правильных посылок всегда гарантируется переход к правильному заключению.

Широкое применение силлогистика нашла также в судебной практике, когда материалы предварительного следствия брались за истинные посылки. Применяя к этим посылкам процедуры порождения новых утверждений по правилам теории Аристотеля, судьи делали вывод о виновности или невиновности подсудимого.

В конце XVI века в алгебре словесная форма записи алгебраических выражений стала тормозить развитие науки и, чтобы облегчить выполнение алгебраических преобразований, была создана буквенная символика, позволяющая выполнять эти преобразования по строго определенным правилам. Заменой силлогизму стала служить более простая и мощная логика первого порядка, а также теория кванторов.

2-й этап - появление **математической или символической логики**. Математическая логика возникла на стыке двух наук: традиционной или философской логики и математики.

- **Определение Символическая логика** — направление в математической логике, изучающее формальные системы. посредством построения формализованных языков.

Основными элементами формализованных языков являются не слова обычных разговорных языков, а некоторые символы, выбираемые и интерпретируемые определённым образом.

Основы символической логики заложил немецкий ученый и философ Готфрид **Вильгельм Лейбниц** (1646-1716). Он попытался построить первые логические исчисления, считал, что можно заменить простые рассуждения действиями со знаками и привел правила. Он создал алгебру высказываний. Но Лейбниц высказал только идею, а развил ее окончательно англичанин Джордж Буль (анг)(1815-1864). Буль считается основоположником математической логики как самостоятельной дисциплины. В его работах в 40-х годах 19 века логика обрела свой алфавит, свою орфографию и грамматику. Недаром начальный раздел математической логики называют алгеброй логики, или булевой алгеброй). Окончательно как раздел математики математическая логика сформировалась в работах Д. Буля (1815 – 1864), Г. Фреге (1848 – 1925), Б. Рассела (1872 – 1970), Д. Гильберта (1862 – 1943).

К концу XIX столетия актуальное значение для математики приобрели вопросы обоснования её основных понятий и идей. Эти задачи имели логическую природу и, естественно, привели к дальнейшему развитию математической логики. В этом отношении показательны работы немецкого математика Г. Фреге (1848 - 1925) и итальянского математика Д. Пеано (1858 - 1932), которые применили математическую логику для обоснования арифметики и теории множеств.

3-й этап связан с XX веком и попытками обосновать справедливость математических доказательств, с исследованиями теории чисел, а также с попыткой разрешить известные логические парадоксы.

1.1 Логические парадоксы

Парадокс, всегда поражает неожиданностью. Наличие парадокса стимулирует к новым исследованиям, более глубокому осмыслению теории, её «очевидных» постулатов и нередко приводит к полному её пересмотру.

Определение Парадоксы - ситуация или высказывание, утверждение, суждение или вывод, которые могут существовать в реальности, но не иметь логического объяснения.

Рассмотрим наиболее известные парадоксы математической логики.

1.2 Парадокс лжеца

Самым знаменитым следует считать *парадокс лжеца*, известный еще со времен глубокой древности. Считается, что этот парадокс был сформулирован древнегреческий философом-идеалистом Евбулидом в IV век до н. э.

По преданию, Эпименид утверждал, что все критяне лжецы. Верно ли это утверждение, если учесть, что сам Эпименид родом с острова Крит?.

Другая простейшая форма этого парадокса: «Говорит ли правду или неправду человек, заявляющий „Я лгу“?».:

«Некто говорит: ”я лгу”.

Если он при этом лжет, то сказанное им есть ложь, и, следовательно, он не лжет. Если же он не лжет, то сказанное им есть истина, и следовательно, он лжет.

В любом случае оказывается, что он лжет и не лжет одновременно.»

1.3 Парадокс Платона и Сократа

На **парадокс лжеца** очень похож и **парадокс Платона - Сократа**.

Когда **Платон** сказал: "*Следующее высказывание **Сократа** будет ложным*", **Сократ** остроумно парировал: "*То, что сказал **Платон**, истинно*".

1.4 Парадокс Рассела

Существует много формулировок этого парадокса. Одна из них традиционно называется *парадоксом брадобрея* и звучит так:

Одному деревенскому брадобрею приказали *«брить всякого, кто сам не бреется, и не брить того, кто сам бреется»*, как он должен поступить с собой? Кто бреет брадобрея?

Парадокс Рассела открыт Бертраном Расселом, демонстрирует противоречивость логической системы, являвшейся ранней попыткой формализации наивной теории множеств Г. Кантора.

1.5 Парадокс о вычислимых функциях

➢ **Определение Вычислимые функции** - множество функций вида $f : N \to N$, которые могут быть реализованы на исполнителе машин Тьюринга.

Легко доказать, что множество всюду определенных вычислимых функций является перечислимым, т. е. их можно перенумеровать в виде последовательности $f_1, f_2, f_3, \ldots$.

Определим теперь новую функцию формулой:

$$g(n) = f_n(n) + 1.$$

Она не входит в нашу последовательность, поскольку при n=1 она отличается от f_1, при n=2 - от f_2 и т. д. Следовательно, она не вычислима.

С другой стороны, ясно, что она вычислима, так как $f_n(n)$ вычислима, а прибавив 1 к $f_n(n)$, мы получим g(n).

1.6 Математическая логика на современном этапе

Развитие математической логики особенно активизировалось в XX нашего века в связи с развитием вычислительной техники и программирования. В настоящее время в связи с бурным развитием информатики, программирования и исследованиями в области искусственного интеллекта значение логики существенно возросло, особенно в прикладном плане.

Областями использования логики в настоящее время являются:

- проектирование цифровых схем;
- исследование семантики языков программирования;
- спецификация, верификация и синтез программ;
- спецификация и верификация параллельных процессов;
- создание логических языков программирования;
- системы искусственного интеллекта

➢ **Определение Математическая логика** - это современная форма логики, которая полностью опирается на формальные математические методы. Она изучает только умозаключения со строго определенными объектами и суждениями, для которых можно однозначно решить, истинны они или ложны.

Математическая логика используется при решении трех групп задач.

Во-первых, это формулировка логических рассуждений с помощью специальных символов и изучение этих рассуждений с использованием математического аппарата.

Во-вторых, это построение формальных теорий (исчислений) для различных математических объектов на основе аксиоматического метода.

В-третьих, это применение аппарата математической логики к различным областям практической деятельности. В настоящее время математическая логика с успехом применяется в радиотехнике, лингвистике, теории автоматического управления, программировании, системах искусственного интеллекта

Задачи для самостоятельного решения

1. Нарисуйте диаграммы Эйлера-Венна, иллюстрирующие суждения:
   1. "Все X являются Y"
   2. "Некоторые X являются Y"
   3. "Ни одно x не является Y"
   4. "Некоторые X не являются Y"
2. Следует ли из того, что "Все X являются Y и некоторые Y являются Z", утверждение "Некоторые X являются Z "?
3. Правильно ли рассуждение, имеющее форму: "Все X являются Y, и некоторые Y являются Z; значит, некоторые Z являются X"?

Контрольные вопросы

1. Что изучает формальная логика?
2. Что изучает математическая логика?
3. Изложите основные этапы развития логики.
4. Области применения математической логики.
5. Перечислите известные парадоксы, основные идеи.
6. Разновидности силлогизмов и модусов: отличие и сходство

Глава 2 Логика высказываний

Логика высказываний является простейшей логикой, максимально близкой к человеческой логике неформальных рассуждений и известна ещё со времён античности.

Логика высказываний (или **пропозициональная логика**, propositional logic or calculus) — это формальная теория, основным объектом которой служит понятие логического высказывания, классическая логика нулевого порядка.

Основным (неопределяемым) понятием математической логики является понятие «простого высказывания».

➢ **Определение Высказывание** - это повествовательное предложение, о котором можно сказать, что оно истинно или ложно (И или Л).

Пример: Земля - планета Солнечной системы. (Истинно); Каждый параллелограмм есть квадрат (Ложно)

Существуют высказывания, о которых нельзя говорить с уверенностью, истинны они или ложны. « Сегодня хорошая погода « (кому-то не очень).

Говорить об истинности или ложности определений бессмысленно. Определение есть соглашение о названии. Например, "Назовем эту музыку гимном". И все тут!..

Пример Высказывание *"Идет дождь"* - простое, а истинное оно или ложное зависит от того, какая погода сейчас за окном. Если действительно не переставая льет дождь, то высказывание - истинное, а если нещадно палит солнце, и бесполезно ждать дождя, то высказывание *"Идет дождь"* будет ложным.

Пример "$x-1=4$" – не высказывание (неизвестно, какие значения принимает x).

"Студент второго курса" не высказывание (не утверждение).

В алгебре логики все высказывания рассматриваются только с точки зрения их логического значения, а от их житейского содержания отвлекаются.

Считается, что каждое высказывание либо истинно, либо ложно и ни одно высказывание не может быть одновременно истинным и ложным.

В дальнейшем будем элементарные высказывания обозначать буквами латинского алфавита: *a,b,c,...,x,y,z,...;* истинное значение – буквой И или цифрой 1, а ложное значение – буквой Л или цифрой 0.

➢ **Определение Элементарные высказывания или простые высказывания** – высказывания, представляющие одно утверждение, не могут быть выражены через другие высказывания.

В дальнейшем будем простые высказывания обозначать строчными латинскими буквами **a, b, c, ..., x, y, z**.

➢ **Определение Составные высказывания или сложные высказывания** – высказывания, которые можно выразить из элементарных высказываний с помощью связок «не», «и», или», «если то», «тогда и только тогда».

Сложные высказывания также обладают одним из двух свойств: «быть истинным» или «быть ложным». При этом истинность или ложность сложного высказывания зависит исключительно от истинности или ложности простых высказываний, из которых они с помощью связок получаются и логической операции используемой в составлении сложного высказывания.

Пример "Число 22 четное" – элементарное высказывание.

Существуют два основных подхода к установлению истинности высказываний: эмпирический (опытный) и логический.

При *эмпирическом подходе* истинность высказывания устанавливается с помощью наблюдений, измерений, проведением экспериментов.

Логический подход заключается в том, что истинность высказывания устанавливается на основе истинности других высказываний, то есть без обращения к фактам, к их содержанию, то есть формально.

Такой подход основан на выявлении и использовании логических связей между высказываниями, входящими в рассуждение.

Алгебра логики не занимается обоснованием того, почему, тому или иному простому (элементарному) высказыванию присваивается значение истинности или ложности, этим занимаются другие разделы математики. Более того алгебра логики отвлекается от смысловой содержательности высказываний, ее интересуют только их значения (истинно или ложно). Такой подход позволяет строить и изучать как угодно сложные (составные) высказывания. Кардинальный сдвиг в анализе стандартных рассуждений произошел в тот период, когда для создания логической теории был применен метод построения формальных систем с помощью специальных символических языков.

Были созданы две мощные формальные системы, которые впервые позволили автоматизировать рассуждения, опирающиеся на схему дедуктивного вывода: исчисления высказываний (ИВ) и исчисления предикатов (ИП).

➢ **Определение Логика высказываний -** раздел логики, в котором вопрос об истинности или ложности высказываний рассматривается и решается на основе изучения способа построения высказываний из *элементарных* (далее не разлагаемых и не анализируемых) высказываний с помощью логических операций конъюнкции ("и"), дизъюнкции ("или"), отрицания ("не"), импликации ("если..., то...") и др.

2.1 Алгебра высказываний

Алгебра высказываний является составной частью одного из современных быстро развивающихся разделов математики – математической логики. Одним из приложений алгебры высказываний является решение логических задач. Исходные объекты алгебры высказываний - это простые (элементарные) высказывания.

Внутри алгебры высказываний не говорится о том, что такое простое высказывание и что такое «истинность» и «ложность».

Вообще, многие математические утверждения можно считать простыми высказываниями при этом принято считать, что они либо истинны, либо ложны, даже если нам неизвестно, каким из двух свойств данное высказывание обладает.

Так, например, «Всякое четное число является суммой двух простых чисел» — высказывание, хотя мы не знаем, каким из двух свойств оно в действительности обладает: это нерешенная проблема Гольдбаха.

Когда речь идет о высказывании нужно иметь в виду следующее:

1. Любое высказывание является либо истинным, либо ложным (закон исключенного третьего).

2. Никакое высказывание не может быть одновременно истинным и ложным (закон противоречия).

3. Предложение, о котором невозможно однозначно решить вопрос, истинно оно или ложно, высказыванием не является.

Истинность или ложность сложных высказывания зависит ни от каких-то внешних причин, а от простых высказываний и логических связок из которых составлено это сложное высказывание.

Из простых высказываний с помощью небольшого числа операций строятся сложные высказывания.

Операции, называемые логическими связками или логическими функциями, примерно соответствуют тому, что в обыденной речи описывается словами «не», «и», «или», «если..., то» и т. п.

Связка полностью может быть описана таблицей, указывающей, какие значения истинности принимает сложное высказывание при различных значениях истинности простых. Такая таблица называется таблицей истинности, соответствующей данной связке.

Введем операции над высказываниями так же, как мы это делали для булевых функций.

➢ **Определение** *Отрицание* высказывания А - высказывание, истинное тогда и только тогда, когда высказывание А ложно.

Читается «не А», или «неверно, что А».

Отрицание определяется следующей таблицей истинности:

A	$\overline{A}$
Л	И
И	Л

Пример Для высказывания «Река Волхов вытекает из озера Ильмень» отрицанием будет высказывание «Неверно, что река Волхов вытекает из озера Ильмень» или «Река Волхов не вытекает из озера Ильмень».

➢ **Определение** *Конъюнкция* двух высказываний A и B - высказывание, истинное тогда и только тогда, когда истинны оба высказывания.

Конъюнкция определяется следующей таблицей истинности. Читается «A и B»

A B	$A \wedge B$
Л Л	Л
Л И	Л
И Л	Л
И И	И

Пример Для высказываний «6 делится на 2», «6 делится на 3» их конъюнкцией будет высказывание «6 делится на 2 и 6 делится на 3», которое, очевидно, истинно.

Для задания двуместных связок удобно записывать матрицы истинности в виде таблиц с двумя входами: строки соответствуют значениям истинности одного элементарного высказывания, столбцы — значениям другого элементарного высказывания, а в клетке пересечения столбца и строки помещается значение истинности соответствующего сложного высказывания.

✓ **Замечание** Связкой $\wedge$ в алгебре логики могут связываться любые, сколь угодно далекие по смыслу высказывания.

Из определения операции конъюнкции видно, что союз «и» в алгебре логики употребляется в том же смысле, что и в повседневной речи. Но в обычной речи не принято соединять

союзом «и» два высказывания, далекие друг от друга по содержанию, а в алгебре логики рассматривается конъюнкция двух любых высказываний.

Например: «*В огороде бузина и в Киеве дядька*».

✓ **Замечание** Из определения операций конъюнкции и отрицания ясно, что высказывание $x \wedge \bar{x}$ всегда ложно.

➢ **Определение** *Дизъюнкция* двух высказываний А и В - высказывание, ложное тогда и только тогда, когда оба высказывания ложны.

Дизъюнкция определяется следующей таблицей истинности. Читается «А или В».

A	B	$A \vee B$
Л	Л	Л
Л	И	И
И	Л	И
И	И	И

Пример Высказывание «В треугольнике АВС угол А или угол В острый истинно, так как обязательно истинно одно из высказываний: «В треугольнике АВС угол А острый», «В треугольнике АВС угол В острый».

✓ **Замечание** В повседневной речи союз «или» употребляется в различном смысле: исключающем и не исключающем.

В алгебре логики союз «или» всегда употребляется в не исключающем смысле.

✓ **Замечание** Из определения операций дизъюнкции и отрицания ясно, что высказывание $x \vee \bar{x}$ всегда истинно.

➢ **Определение** *Импликация* двух высказываний А и В - высказывание, ложное тогда и только тогда, когда А - истинно, а В - ложно.

Высказывание А называется *посылкой* (гипотезой, антецедентом) импликации, а высказывание В - *заключением* (выводом, консеквентом) импликации.

Импликации соответствуют следующие выражения разговорной речи: “*A* влечет за собой *B*”; или “из *A* следует *B*”; или “если *A*, то *B*”.

Импликация определяется следующей таблицей истинности:

A	B	$A \to B$
Л	Л	И
Л	И	И
И	Л	Л
И	И	И

Пример Истинными являются следующие импликации: “Если в доме 5 этажей, то Иванов живет в квартире 50”;

“Если идет снег, то $2 \times 2 = 5$.

Употребление слов “если…, то…” в алгебре логики отличается от употребления их в обыденной речи, где как правило, считаем, что, если высказывание x ложно, то высказывание “Если x, то y” вообще не имеет смысла.

Кроме того, строя предложение вида “ если x, то y” в обыденной речи, всегда подразумеваем, что предложение y вытекает из предложения x.

Употребление слов “если…, то…” в математической логике **не требует этого**, поскольку в ней смысл содержания высказываний не рассматривается.

Импликация играет важную роль в математических доказательствах, так как многие теоремы формулируются в условной форме “Если x, то y”.

➢ **Определение** *Эквивалентность(эквиваленция)* двух высказываний A и B - высказывание, истинное тогда и только тогда, когда истинностные значения A и B совпадают.

Говорят, что A эквивалентно B или A имеет место тогда и только тогда, когда имеет место B.

Читается «A тогда и только тогда, когда B»

Эквивалентность определяется следующей таблицей истинности:

A	B	$A \sim B$
Л	Л	И
Л	И	Л
И	Л	Л
И	И	И

Эквивалентность играет большую роль в математических доказательствах. Известно, что значительное число теорем формулируется в форме необходимых и достаточных условий, т.е. в форме эквивалентности. В этом случае, зная об истинности или ложности одного их двух членов эквивалентности и доказав истинность самой эквивалентности, делаем заключение об истинности или ложности второго члена эквивалентности.

Высказывания вместе с определенными для них операциями образуют алгебру высказываний.

➢ **Определение** Символы отрицание, конъюнкция, дизъюнкция, импликация и эквивалентность называются **пропозициональными связками** или связками исчисления высказываний.

2.2 Приоритет или ранг связок

С помощью логических операций над высказываниями из заданной совокупности высказываний можно строить различные сложные высказывания. При этом порядок выполнения операций указывается скобками. Например, из трёх высказываний x,y,z можно построить высказывания

$(x \wedge y) \vee \bar{z}$ и $x \to \overline{(y \vee (x \wedge z))}$. Если скобки не указаны, то приоритет связок устанавливается в виде:

$$\neg, \quad \wedge, \vee, \to, \leftrightarrow .$$

Скобки можно опускать, придерживаясь следующего порядка действий: конъюнкция выполняется раньше, чем все остальные операции, дизъюнкция выполняется раньше, чем импликация и эквивалентность.

Если над формулой стоит знак отрицания, то скобки тоже опускаются.

Пример Формулы $(x \wedge y) \vee \bar{z}$ и $x \to \overline{(y \vee (x \wedge z))}$ могут быть написаны так: $x \wedge y \vee \bar{z}$ и $x \to \overline{y \vee x \wedge z}$, а также $xy \vee \bar{z}$ и $x \to \overline{y \vee xz}$.

2.3 Исчисление высказываний

Давая описание алгебры высказываний, пользовались логическими значениями высказываний (истина, ложь). Но понятия истинности и ложности не математические. В связи с этим желательно построить математическую логику, не пользуясь понятиями истинности и ложности. Необходимо также при этом построении не применять самих законов логики.

При формализации математической теории полностью отвлекаются от ее содержания. Теоремы воспринимаются просто как формулы, которые могут быть выведены по определенным правилам. Поэтому формальные теории иначе называют исчислениями.

Если имеется несколько высказываний, то при помощи логических операций можно образовывать различные новые высказывания.

Пример Рассмотрим простые высказывания.

A = "Будет холодное лето".

B = "Будет дождливое лето".

C = "Будет засушливое лето".

D = "Будет хороший урожай".

Формула $(A \wedge B \vee C) \to \overline{D}$ соответствует сложному высказыванию: "Если будет холодное и дождливое или засушливое лето, урожай будет плохим".

➢ **Определение Исчисление высказываний** – это аксиоматическая логическая система, интерпретацией которой является алгебра высказываний.

Наибольший интерес представляет построение формальной системы, которая среди всех возможных высказываний выделяет такие, которые являются логическими законами (правильно построенными рассуждениями, логическими умозаключениями, тавтологиями, общезначимыми высказываниями).

Формальные теории, не пользуясь естественным (разговорным) языком, нуждаются в собственном формальном языке, на котором записываются встречающиеся в нем выражения.

➢ **Определение** Формальная система, порождающая высказывания, которые являются тавтологиями и только их, называются **исчислением высказываний** (ИВ).

Формальная система ИВ определяется:

$$ИВ = \langle \{аксиомы\}, ПВ - правила\ вывода \rangle$$

➧ *Какие символы лучше использовать для обозначения логических связок?* Остановимся на следующих обозначениях: отрицание, конъюнкция, дизъюнкция, импликация и эквивалентность. Обычно логические значения результатов применения связок записываются в виде таблиц (т.н. таблицы истинности).

2.4 Формулы логики высказываний

Формулы алгебры высказываний осмысленные выражения, полученные из символов элементарных высказываний, символов высказательных переменных, знаков операций (конечные числа) и скобок, определяющих порядок действий.

➢ **Определение Алфавит исчисления высказываний** – любое непустое множество, элементы которого есть символы трех категорий:

1. Символы первой категории: $x, y, z, \ldots, x_1, x_2, \ldots$ Эти символы будем называть *переменными высказываниями.* – буквы латинского алфавита с индексом или без него.
2. Символы второй категории: $\vee, \wedge, \rightarrow, -$. они носят общее название **логических связок.** Первый из них – знак дизъюнкции или логического сложения, второй – знак конъюнкции или логического умножения, третий – знак импликации или логического следования и четвертый – знак отрицания.
3. Третью категорию составляет пара символов ( ), называемая скобками или разделитель.

Других символов исчисление высказываний не имеет

➢ **Определение. Формула** – правильно построенная составное высказывание

1) Всякая буква есть **формула**.

2) Если A, B - формулы, то формулами являются также $\neg A$, $A \vee B$, $A \wedge B$, $A \to B$, $A \leftrightarrow B$.

Одновременно с понятием формулы вводится понятие подформулы или части формулы.

➢ **Определение Подформулой** элементарной формулы является

1. она сама формула.
2. Если формула имеет вид $\overline{A}$, то ее подформулами являются: она сама, формула А и все подформулы формулы А.
3. Если формула имеет вид (А*В)(здесь и в дальнейшем под символом * будем понимать любой из трех символов $\vee, \wedge, \to$), то ее подформулами являются: она сама, формулы А и В и все подформулы формул А и В.

Формулы алгебры логики обозначаются большими буквами латинского алфавита А,В,С…

➢ **Определение** Формула называется **тавтологией,** если она принимает только истинные значения при любых значениях букв.

➢ **Определение** Формула ложная при любых значениях букв называется **противоречием.**

➢ **Определение** Формула называется **выполнимой,** если на некотором наборе распределения истинностных значений переменных она принимает значение И.

➢ **Определение** Формула называется **опровержимой**, если при некотором распределении истинностных значений переменных она принимает значение Л.

Пример

$(x \wedge y), (x \vee z), (y \to z), \overline{x}$ являются формулами

Будут формулами слова:

$\overline{(x \wedge y)}, ((x \vee z) \wedge (y \to z)), ((x \wedge y) \to (y \to z))$.

Пример

Для формулы $\left((x \wedge \overline{y}) \to \overline{(z \vee y)}\right)$ ее подформулами будут:

$\left((x \wedge \overline{y}) \to \overline{(z \vee y)}\right)$ - подформула нулевой глубины,

$(x \wedge \overline{y}), \overline{(z \vee y)}$ -подформулы первой глубины,

$x, \overline{y}, (\overline{z} \wedge y)$ -подформулы второй глубины,

$y, \overline{z}$ -подформулы третьей глубины,

z -подформула четвертой глубины.

Не являются формулами слова: $\wedge x, (x \wedge y, x \vee y,$ поскольку одни из них не взяты в скобки, в других скобка лишь одна.

Таким образом, по мере "погружения вглубь структуры формулы" выделяем подформулы все большей глубины.

Будем опускать в записи формул скобки по тем же правилам, что и в алгебре высказываний.

В связи с этими правилами формулы

$$((A \wedge B) \vee C), \overline{(A \vee B)}, ((A \vee B) \to (C \wedge D))$$

будем писать

$A \wedge B \vee C, \overline{A \vee B}, A \vee B \to C \wedge D$ соответственно.

➢ **Определение** Конкретный набор истинностных значений, приписанных переменных называется **интерпретацией** формулы.

Формула может быть истинной при одной интерпретации и ложной при другой.

➢ **Определение Язык исчисления высказываний** – это исходные символы, из которых строятся формулы исчисления высказываний.

➢ **Определение** Две формулы A и B называются **логически эквивалентными** или **равносильными**, если они принимают одинаковые логические значения при любом наборе значений входящих в формулы элементарных высказываний или тогда и только тогда, когда формула $A \leftrightarrow B$ – тавтология.

Равносильность формул будем обозначать знаком ≡, а запись А ≡В означает, что формулы А и В равносильны.

❖ **Теорема**. Отношение логической эквивалентности-отношение эквивалентности (Рефлексивно, симметрично, транзитивно).

Между понятиями равносильности и эквивалентности существует следующая связь: если формулы А и В равносильны, то формула А↔В – тавтология, и обратно, если формула А↔В – тавтология, то формулы А и В равносильны.

Из курса дискретной математики известны основные логические эквивалентности (равносильности), которые являются примерами тавтологий.

Все логические законы должны быть тавтологиями. Иногда законы называются **правилами вывода,** которые определяют правильный вывод из посылок.

2.5 Законы алгебры высказываний

Наиболее простые и необходимые истинные связи между высказываниями выражаются в основных законах формальной логики. Эти законы являются основными потому, что в логике они играют особо важную роль, являются наиболее общими. Они позволяют упрощать логические выражения и строить умозаключения и доказательства. Рассмотрим основные законы подробнее.

1. **Коммутативность**:
$$A \vee B = B \vee A,\ A \wedge B = B \wedge A.$$
2. **Ассоциативность:**
$$A \vee (B \vee C) = (A \vee B) \vee C,$$
$$A \wedge (B \wedge C) = (A \wedge B) \wedge C.$$
3. **Дистрибутивность:**
$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C),$$
$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C).$$
4. **Идемпотентность**:
$$A \vee A = A,\ A \wedge A = A.$$
5. **Закон двойного отрицания**: $\neg\neg A = A.$
6. **Закон исключения третьего**: $A \vee \neg A = 1.$
7. **Закон противоречия**: $A \wedge \neg A = 0.$
8. **Законы де Моргана**:
$$\neg(A \vee B) = \neg A \wedge \neg B,\ \neg(A \wedge B) = \neg A \vee \neg B.$$

9. **Законы с логическими константами**

В алгебре логики нет показателей степеней и коэффициентов. Конъюнкция одинаковых "сомножителей" равносильна одному из них

$$A \vee 1 = 1,\ A \vee 0 = A,\ A \wedge 1 = A,\ A \wedge 0 = 0.$$

Здесь A, B и C – любые буквы.

Примеры Формула $A \wedge B \to A$ тавтология.

2. $(A \wedge B) \wedge C \leftrightarrow A \wedge (B \wedge C)$ тавтологией.

3. $(A \vee B) \vee C \leftrightarrow A \vee (B \vee C)$ тавтологией.

Теорема. Пусть формулы A и $A \to B$ – тавтологии. Тогда формула B – тавтология.

Теорема. Пусть формула A – тавтология, P_1, P_2, ..., P_k – буквы в формуле A, A_1, A_2, ..., A_k – любые формулы. Тогда новая формула $B = A(A_1, A_2, \ldots, A_k)$ – тавтология.

Равносильности, выражающие одни логические операции через другие

1. $x \leftrightarrow y \equiv (x \to y) \wedge (y \to x)$
2. $x \to y \equiv \bar{x} \vee y$.
3. $\overline{x \wedge y} \equiv \bar{x} \vee \bar{y}$.
4. $\overline{x \vee y} \equiv \bar{x} \wedge \bar{y}$.
5. $\overline{\bar{x} \wedge \bar{y}} \equiv x \vee y$.
6. $\overline{\bar{x} \vee \bar{y}} \equiv x \wedge y$.

Здесь 3, 4, 5, 6 – законы Моргана.

Ясно, что равносильности 5 и 6 получаются из равносильностей 3 и 4, соответственно, если от обеих частей последних взять отрицания и воспользоваться законом снятия двойного отрицания.

Таким образом, в доказательстве нуждаются первые четыре равносильности. Докажем одну из них: первую

Так как при одинаковых логических значениях x и y истинными являются формулы $x \leftrightarrow y, x \to y, y \to x$, то истинной будет и конъюнкция $(x \to y) \wedge (y \to x)$. Следовательно, в этом случае обе части равносильности имеют одинаковые истинные значения.

Пусть теперь x и y имеют различные логические значения. Тогда будут ложными эквивалентность $x \leftrightarrow y$ и одна из двух импликаций $x \to y$ или $y \to x$. Но при этом будет ложной и конъюнкция $(x \to y) \wedge (y \to x)$.

Таким образом, и в этом случае обе части равносильности имеют одинаковые логические значения.

Аналогично доказываются равносильности 2 и 4.

Из равноситльностей этой группы следует, что всякую формулу алгебры логики можно заменить равносильной ей формулой, содержащей только две логические операции: конъюнкцию и отрицание или дизъюнкцию и отрицание.

Дополнительные законы

1. Закон склеивания (расщепления)

$xy \vee \bar{x}y \equiv y$, $xy \vee x\bar{y} \equiv x$;

$(x \vee y) \wedge (\bar{x} \vee y) \equiv y$, $(x \vee y) \wedge (x \vee \bar{y}) \equiv x$.

2. Законы поглощения

$x \vee xy \equiv x$; $x(x \vee y) \equiv x$.

3. Закон Блейка- Порецкого

$x \vee \bar{x}y \equiv x \vee y$.

4. Закон свертки логического выражения

$xy \vee \bar{x}z \vee yz \equiv xy \vee \bar{x}z$.

5. Закон двойственности

➢ **Определение** Формулы A и A^* называются **двойственными**, если формула A^* получается из формулы A путем замены в ней каждой операции на двойственную.

Имеет место следующий закон двойственности: если формулы A и B равносильны, то равносильны и им двойственные формулы, т.е. $A^* \equiv B^*$.

2.6 Проблема разрешимости для логики высказываний

Проблема разрешимости для любого аксиоматического исчисления, в том числе и для исчисления высказываний, состоит в существовании алгоритма, который по любой формуле устанавливает, является она в этом исчислении доказуемой или не является. Если такого алгоритма не существует, то аксиоматическое исчисление алгоритмически неразрешимо.

Исчисление высказываний алгоритмически разрешимо. Разрешающий алгоритм состоит в проверке для данной формулы, является ли она тождественно истинной. Если да, то формула доказуема в ИВ; если нет, то формула в ИВ не доказуема.

Проблемой разрешимости для логики высказываний называют также следующую проблему: существует ли алгоритм, который позволил бы для произвольной формулы в конечное число шагов определить, является ли она тавтологией?

❖ **Теорема**. Формула является тавтологией в том и только том случае, если в ее КНФ в любую из элементарных дизъюнкций в качестве дизъюнктивных членов входит какая-нибудь переменная и ее отрицание.

Двойственное утверждение справедливо и для противоречия.

❖ **Теорема**. Формула является противоречием в том и только том случае, если в ее ДНФ каждая элементарная конъюнкция одновременно содержит в качестве конъюнктивных членов какую-нибудь переменную и ее отрицание

Следовательно, приведя формулу равносильными преобразованиями к КНФ, можно установить, является ли она тождественно-истинной, а приведя ее к ДНФ, можно установить, является ли она тождественно-ложной.

Очевидно, что проблема разрешимости алгебры логики разрешима.

Действительно, для каждой формулы алгебры логики может быть составлена таблица истинности, которая и даст ответ на поставленный вопрос. Однако практическое использование

таблиц истинности при большом количестве переменных x_i затруднительно.

Существует другой способ, позволяющий, не используя таблицы истинности, определить, к какому классу относится формула F. Этот способ основан на приведении формулы к нормальной форме (ДНФ или КНФ) и использовании алгоритма, который позволяет определить, является ли рассматриваемая формула тождественно истинной или не является. Одновременно с этим решается вопрос о том, будет ли формула А выполнимой.

Применим критерий тождественной истинности к формуле A. Если окажется, что формула A – тождественно истинная, то задача решена. Если же окажется, что формула A не тождественно истинна, то применим критерий тождественной истинности к формуле $\overline{A}$. Если окажется, что формула $\overline{A}$ - тождественно истинная, то ясно, что формула A - тождественно ложная, и задача решена. Если же формула $\overline{A}$ не тождественно истинная, то остаётся единственный возможный результат: формула A выполнима.

Установим теперь критерий тождественной истинности произвольной формулы алгебры логики. С этой целью предварительно сформулируем и докажем критерий тождественной истинности элементарной дизъюнкции.

Контрольные вопросы

1. Что такое высказывание?
2. Какие высказывания бывают?
3. Какие высказывания называются простыми, а какие сложными?
4. Что не является высказыванием?
5. Основные логические операции и их свойства, перечислите.
6. Отрицание высказывания. Конъюнкция двух высказываний. Дизъюнкция двух высказываний.

7. Импликация двух высказываний. Эквивалентность двух высказываний.
8. Союзы языка и логические операции (язык и логика). Общий взгляд на логические операции.
9. Конструирование сложных высказываний.
10. Понятие формулы алгебры высказываний.
11. Логическое значение составного высказывания.
12. Классификация формул алгебры высказываний.

Глава 3 Формальные теории

Одной из фундаментальных идей является идея формализации теорий, то есть последовательного проведения аксиоматического метода построения теории. При этом не допускается использовать любые предположения об объектах теории, кроме тех, которые выражены явно в виде аксиом.

Аксиомы рассматривают как формальные последовательности символов (выражения, формулы или слова), а методы доведения - как методы получения одних выражений из других с помощью операций над символами.

Такой формальный подход гарантирует четкость и однозначность исходных (начальных) утверждений и корректность, однозначность вывода.

➢ **Определение** *Формальная теория* считается заданной, если известны следующие четыре составляющих:

1. *Алфавит* – конечное или счетное множество символов.
2. *Формулы*, которые по специальным правилам строятся из символов алфавита.
3. *Аксиомы*
4. *Правила вывода* – множество отношений, позволяющие из аксиом получать теоремы формальной теории.

➢ **Определение Вывод** формальной теории - последовательность формул A_1, A_2, …, A_n, в которой все формулы – либо аксиомы, либо получаются из предыдущих по правилам вывода.

➢ **Определение** Формула A **выводима** из множества формул Γ ($\Gamma \vdash A$), если существует вывод A_1, A_2, …, A_n, где $A_n = A$, и есть три возможности: $A_i \in \Gamma$; A_i - аксиома; A_i получаются из предыдущих формул по правилам вывода.

Формулы из множества Γ называются *посылками* или *гипотезами* вывода.

Важной особенностью формальных теорий является то, что содержательные утверждения заменены в них последовательностями символов, маннипуляции с которыми

основываются лишь на их внешнем виде, и подразумеваемая логическая система явным образом включается в теорию.

➢ **Определение** Аксиоматические теории, в которых правила логики явно не заданы, называются **неформальными** или *интуитивными*

➢ **Определение** Формула называется **общезначимой**, если она истинна в любой интерпретации.

Общезначимые формулы занимают важнейшие места в формальной логике, в том числе в алгебры высказываний, в логике высказываний играют ту же роль, что и тавтологии, истинны в силу своей структуры, независимо от истинностных значений составляющих их формул.

➧ **Пример** Для любой формулы A, формальная схема $A \vee \overline{A}$ принимает значение 1 независимо от значения формулы A. Рассмотренная формальная схема $A \vee \overline{A}$ выражает один из основных законов формальной логики, закон исключенного третьего.

➢ **Пример** "Если книга сложная, то она неинтересная. Эта книга интересная. Значит, она несложная".

Введем высказывания: А = "Книга сложная"; В = "Книга интересная". Схема рассуждения имеет вид:

$$\frac{A \supset \neg B,\ B}{\neg A}$$

Докажем, что формула ((А ⊃ ¬В) & В) ⊃ ¬А является тождественно-истинной.

((А ⊃ ¬В)&В) ⊃ ¬А ≡ ¬((А ⊃ ¬В)& В) ∨ ¬А ≡ (А & В) ∨¬В∨ ¬А ≡ (¬А ∨ ¬В ∨ А)&(¬А∨ ¬В ∨ В) ≡ И.

Значит, рассуждение правильное.

➢ **Пример** "Если будет хорошая погода, я пойду гулять. Если будет плохая погода, я буду читать книгу. Погода будет хорошая. Следовательно, я не буду читать книгу".

Введем высказывания: А = "Будет хорошая погода"; В = "Я пойду гулять". С = "Я буду читать книгу". Схема рассуждения имеет вид:

$$\frac{A \supset B,\ \neg A \supset C,\ A}{}$$

$\neg C$

Найдем КНФ формулы $((A \supset B) \& (\neg A \supset C) \& A) \supset \neg C$:

$((A \supset B) \& (\neg A \supset C) \& A) \supset \neg C \equiv \neg((A \supset B) \& (\neg A \supset C) \& A) \vee \neg C \equiv \neg(A \supset B) \vee \neg(\neg A \supset C) \vee \neg A) \vee \neg C \equiv A \& \neg B \vee \neg A \& \neg C \vee \neg A \vee \neg C \equiv A \& \neg B \vee \neg A \vee \neg C \equiv (A \vee \neg A \vee \neg C) \& (\neg B \vee \neg A \vee \neg C) \equiv \neg B \vee \neg A \vee \neg C$.

Полученная КНФ формулы не содержит одновременно какой-либо переменной и ее отрицания. Следовательно, формула не является тождественно-истинной, а рассуждение не является правильным.

Этапом в построении исчисления высказываний является выделение класса доказуемых (выводимых) формул. Сначала определяются исходные доказуемые выводимые формулы (аксиомы), а затем определяются правила вывода, которые позволяют из имеющихся доказуемых формул получить новые доказуемые формулы.

3.1 Аксиоматический метод

Математическая наука достигает совершенства лишь тогда, когда она принимает характер аксиоматической теории. Назначение аксиоматического метода состоит в ограничении произвола при принятии научных суждений в качестве истин данной теории. Построение науки на основе аксиоматического метода обычно называется *дедуктивным*.

➢ **Определение Аксиоматический метод -** способ построения научной теории, при котором в её основу кладутся некоторые исходные положения (суждения) — *аксиомы,* из которых все остальные утверждения этой науки должны выводиться чисто логическим путём, посредством доказательств.

Все понятия дедуктивной теории вводятся посредством определений, выражающих их через ранее введённые понятия.

В той или иной мере дедуктивные доказательства, характерные для аксиоматического метода, применяются во многих науках. Главной областью его приложения до сих пор остаются математика и символическая логика, а также

некоторые разделы физики (механика, термодинамика, электродинамика и др.).

В аксиоматическом построении математической теории предварительно выбирается некоторая система неопределяемых понятий и отношения между ними.

Эти понятия и отношения называются основными. Далее без доказательства принимаются основные положения рассматриваемой теории – аксиомы. Всё дальнейшее содержание теории выводится логически из аксиом. Впервые аксиоматическое построение математической теории было предпринято Евклидом в построении геометрии.

Изложение этой теории в "Началах" Евклида не безупречно. В доказательстве теорем используются нигде явно не сформулированные положения, которые считаются очевидными.

Таким образом, в этом построении отсутствует необходимая логическая строгость, хотя истинность всех положений теории не вызывает сомнений.

Отметим, что такой подход к аксиоматическому построению теории оставался единственным до XIX века. Большую роль в изменении такого подхода сыграли работы Н. И. Лобачевского.

Лобачевский впервые в явном виде высказал убеждения в невозможности доказательства пятого постулата Евклида и подкрепил это убеждение созданием новой геометрии.

Одна из формулировок пятого постулата гласит, - «На плоскости через точку, находящуюся вне прямой, можно провести только одну прямую, параллельную данной».

В планиметрии Лобачевского через точку вне прямой на плоскости можно провести сколько угодно различных прямых, параллельных данной. Это – аксиома.

В планиметрии Римана через точку вне прямой нельзя провести ни одной прямой, параллельной данной. И это утверждение тоже аксиоматизируется.

Позже немецкий математик Ф. Клейн (1849-1925) доказал непротиворечивость геометрии Лобачевского, чем фактически была доказана и невозможность доказательства пятого постулата Евклида.

В работах Н. И. Лобачевского и Ф. Клейна впервые в истории математики проблемы невозможности доказательства и непротиворечивости в аксиоматической теории.

Аксиоматический метод прошёл в своём историческом развитии **3 стадии**.

Первая связана с построением геометрии в Древней Греции. Основное сочинение этого периода — «Начала» Евклида (хотя, и до него Пифагор, которому приписывается открытие аксиоматического метода, а затем Платон и его ученики немало сделали для развития геометрии на основе аксиоматического метода).

В то время считалось, что в качестве аксиом должны выбираться суждения, истинность которых «самоочевидна», так что истинность теорем считалась гарантированной безупречностью самой логики. Но Евклиду не удалось ограничиться чисто логическими средствами при построении геометрии на основе аксиом.

Во времена Евклида обращения к интуиции могли и не восприниматься как выход за пределы логики — прежде всего потому, что сама логика не была ещё аксиоматизирована. Не было и достаточной отчётливости во введении первоначальных понятий и при определении новых понятий.

Вторая стадия в истории аксиоматического метода связана с открытием Н. И. Лобачевским возможности построить непротиворечивым образом геометрию, исходя из систем аксиом, отличной от евклидовой. Это открытие разрушило убеждение в абсолютной («очевидной» или «априорной») истинности аксиом и основанных на них научных теорий.

Аксиомы стали пониматься просто как исходные положения данной теории, вопрос же об их истинности в том или ином смысле (и выбор в качестве аксиом) выходит за рамки аксиоматической теории как таковой и относится к её взаимоотношению с фактами, лежащими вне её.

Появилось много (и притом различных) геометрических, арифметических и алгебраических теорий, которые строились средствами аксиоматического метода (работы Р. Дедекинда, Г. Грасмана и др.). Эта стадия развития аксиоматического метода

завершилась созданием аксиоматических систем арифметики (Дж. Пеано, 1891), геометрии (Д. Гильберт, 1899), исчисления высказываний и предикатов (А. Н. Уайтхед и Б. Рассел, Англия, 1910) и аксиоматической теории множеств (Э. Цермело, 1908).

Гильбертовская аксиоматизация геометрии позволила Ф. Клейну и А. Пуанкаре доказать непротиворечивость геометрии Лобачевского относительно евклидовой геометрии посредством указания интерпретации понятий и предложений неевклидовой геометрии в терминах геометрии Евклида, или, как говорят, построения модели первой средствами второй.

Метод моделей (интерпретаций) стал с тех пор важнейшим методом установления относительной непротиворечивости аксиоматических теорий.

В то же время со всей отчётливостью выявилось, что, кроме «естественной» интерпретации у аксиоматической теории могут быть и др. интерпретации, причём её можно с равным основанием считать «говорящей» о каждой из них.

Последовательное развитие этой идеи и стремление точно описать логические средства вывода теорем из аксиом привели Гильберта к концепции формального аксиоматического метода, характерной для третьей, современной его стадии.

Третья стадия — полная формализация языка науки, при которой её суждения рассматриваются просто как последовательности знаков (формулы), не имеющие как таковые никакого смысла (который они приобретают лишь при некоторой конкретной интерпретации). Это относится и к аксиомам — как общелогическим, так и специфическим для данной теории. Для вывода теорем из аксиом (и вообще одних формул из других) формулируются специальные правила вывода (например, т. н. правило modus ponens — «правило зачёркивания», позволяющее получить *B* из *A* и «A влечёт B»).

Доказательство в такой теории это просто последовательность формул, каждая из которых либо есть аксиома, либо получается из предыдущих формул последовательности по какому-либо правилу вывода. В рамках созданной Гильбертом теории доказательств, можно было бы доказать непротиворечивость и полноту всей классической

математики. Несмотря на ряд значительных результатов в этом направлении, гильбертовская программа в целом (её обычно называют формализмом) невыполнима, т. к., согласно важнейшему результату К. Гёделя (1931), всякая достаточно богатая непротиворечивая формальная система непременно неполна (теорема о неполноте). Оказывается в сложной аксиоматической системе существуют формулы, которые нельзя ни доказать, ни опровергнуть.

3.2 Теорема Гёделя о неполноте

Теорема Гёделя свидетельствует об ограниченности аксиоматического метода (хотя определённые расширения допускаемых метатеоретических средств и позволили немецкому математику Г. Генцену, П. С. Новикову и др. математикам получить доказательство непротиворечивости формализованной арифметики).

Курт Фридрих Гёдель (1906—1978) — австрийский логик, математик. Основные труды по теории множеств, математической логике и теории относительности. Автор теорем о невозможности полной формализации всей существующей математики и доказательства ее непротиворечивости. Результаты Геделя устанавливают ограниченность формальных методов.

- **Теорема Геделя (о неполноте)** Всякая система математических аксиом начиная с определенного уровня сложности либо внутренне противоречива, либо неполна
- Любая непротиворечивая формальная теория, включающая арифметику целых чисел, неполна.

Каким же образом доказывается теорема Геделя о неполноте формальных систем?

Идея доказательства заключается в том, чтобы построить пример формулы, которая была бы недоказуема и, вместе с тем, содержательно истинна. Таковой являлась бы формула, содержательный смысл которой заключается в том, что она утверждает свою собственную недоказуемость, т.е.

невыводимость из аксиом рассматриваемой формальной системы.

Возьмем любое утверждение типа: «Предположение № 1 в данной системе аксиом логически недоказуемо» и назовем его «утверждением А». Так вот, Гёдель попросту доказал следующее удивительное свойство *любой* системы аксиом:

«Если можно доказать утверждение А, то можно доказать и утверждение не-А».

Иными словами, если можно доказать справедливость утверждения «предположение 1 *недоказуемо*», то можно доказать и справедливость утверждения «предположение 1 *доказуемо*». Если система аксиом полна (то есть любое утверждение в ней может быть доказано), то она противоречива.

Единственным выходом из такой ситуации остается принятие неполной системы аксиом. То есть, останутся утверждения «типа А», которые являются заведомо истинными или ложными, — и можем судить об их истинности лишь *вне* рамок принятой аксиоматики. Если же таких утверждений не имеется, значит, аксиоматика противоречива, и в ее рамках неизбежно будут присутствовать формулировки, которые можно одновременно и доказать, и опровергнуть.

Первая слабая теорема Гёделя о неполноте: «Любая формальная система аксиом содержит неразрешенные предположения».

Вторая, или *сильная теорема Гёделя о неполноте*: «Логическая полнота (или неполнота) любой системы аксиом не может быть доказана в рамках этой системы. Для ее доказательства или опровержения требуются дополнительные аксиомы (усиление системы)».

Английский математик и физик Роджер Пенроуз (Roger Penrose, р. 1931) показал, что теоремы Гёделя можно использовать для доказательства наличия принципиальных различий между человеческим мозгом и компьютером.

Компьютер действует строго логически и не способен определить, истинно или ложно утверждение А, если оно выходит за рамки аксиоматики, а такие утверждения, согласно теореме Гёделя, неизбежно имеются.

Человек же, столкнувшись с таким логически недоказуемым и неопровержимым утверждением А, всегда способен определить его истинность или ложность — исходя из повседневного опыта. Человеческий мозг способен понять всю глубину истины, заключенной в теоремах Гёделя, а компьютерный — никогда.

Пример Существование Бога недоказуемо! Иначе это была бы теорема.

Это только кажется, что аксиоматические системы - это сложно. Любой может придумать их сколько угодно. Более простым делом вам вряд ли приходилось заниматься.

Пример В качестве языка можно объявить любые "слова" из последовательности буквы Я.

- Букву Я объявим аксиомой.
- Правило вывода будет удваивать букву Я.

То есть придумана теория, в которой выводимы любые последовательности (слова), состоящие из буквы Я.

Я ЯЯ ЯЯЯ ... ЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯЯ ...

И все бы хорошо, только такая строго заданная теория мало что дает создателю, кроме радости созидания.

Теория, созданная из буквы Я, не привязана к понятию истинности. Поэтому она бессмысленна, как бессистемная перестановка детских кубиков.

3.3 Непротиворечивость аксиоматической теории

Непротиворечивость аксиоматической теории является одним из основных требований, предъявляемых к системе аксиом данной теории. Она означает, что из данной системы аксиом нельзя логическим путём вывести два противоречивых друг другу утверждения. Противоречивые модели математику не интересуют, поскольку в них чисто логическим путем можно доказать все, что угодно

➢ **Определение** Исчисление высказываний называется **непротиворечивым**, если в нем существует такая формула A, что не доказуема и сама формула и ее отрицание.

➢ **Теорема** Исчисления высказываний противоречиво тогда, и только тогда, когда в нем доказуема любая формула исчисления высказываний.

Доказательство непротиворечивости аксиоматических теорий можно осуществить различными методами. Одним из них является *метод моделирования или интерпретаций*. Здесь в качестве основных понятий и отношений выбираются элементы некоторого множества и отношения между ними, а затем проверяется, будут ли выполняться для выбранных понятий и отношений аксиомы данной теории, то есть строится модель для данной теории.

➢ **Определение Интерпретация теории -** приписывание значений первичным понятиям аксиоматической теории.

Если некоторая совокупность объектов и соответствий между ними, выбранных в качестве значений первоначальных понятий аксиоматической теории, то есть в качестве ее интерпретации, удовлетворяет всем аксиомам теории, то она называется *моделью* данной аксиоматической теории (или моделью системы аксиом теории.

Так, аналитическая геометрия является арифметической интерпретацией геометрии Евклида. Ясно, что метод моделирования сводит вопрос о непротиворечивости одной теории к проблеме непротиворечивости другой теории.

В тех случаях, когда для некоторой системы аксиом удается подобрать конкретные объекты и конкретные соотношения между ними так, что все аксиомы выполняются, говорят, что найдена *интерпретация* (или *модель*) данной системы аксиом.

Алгебра логики является интерпретацией булевой алгебры. Алгебра Буля имеет и другие интерпретации. Например, если под основными элементами $x,y,z,\ldots$множества М подразумевать множества, под операциями “+”, “·”, “–“ объединение, пересечение, дополнение соответственно, а под знаком равенства – знак равенства множеств, то мы приходим к алгебре

множеств. Нетрудно убедиться, что в алгебре множеств все аксиомы алгебры Буля выполняются.

Непротиворечивость евклидовой геометрии никогда не была доказана, хотя почти все "уверены" в ее непротиворечивости.

Доказательство ее относительной непротиворечивости может быть получено с помощью интерпретации, при которой точки интерпретируются посредством упорядоченных пар действительных чисел (x, y); а прямые - уравнениями первой степени $ax + by + c = 0$.

Наличие модели построенной с помощью системы действительных чисел доказывает относительную непротиворечивость евклидовой геометрии: она непротиворечива, если непротиворечива теория действительных чисел.

Но непротиворечивость теория действительных чисел также до сих пор не доказана.

В настоящее время непротиворечивость многих областей классической математики сведена к непротиворечивости арифметики. Тем не менее, "абсолютная" непротиворечивость ни евклидовой геометрии, ни теории действительных чисел, ни арифметики натуральных чисел не установлена. Уверенность в непротиворечивости этих теорий дает практика.

Контрольные вопросы

1. Понятие аксиоматической теории.
2. Как возникают аксиоматические теории. Примеры аксиоматических теорий.
3. Интерпретации и модели аксиоматической теории
4. Свойства аксиоматических теорий.
5. Непротиворечивость. Категоричность. Независимость системы аксиом. Полнота.
6. Понятие формальной аксиоматической теории.
7. Язык и теоремы формальной теории.
8. Формализация теории аристотелевых силлогизмов.
9. Проблемы полноты, разрешимости и непротиворечивости исчисления высказываний.

Глава 4 Система аксиом исчисления высказываний

В основу исчисления высказываний могут быть положены различные системы аксиом, эквивалентные между собой в том смысле, что определяемый ими класс выводимых формул – один и тот же.

На сегодняшнее время известно ≈20 ИВ, которые отличаются друг от друга аксиомами (схемами аксиом) и правилами выводов.

Пример ИВ Уйтхеда и Рассела (1920÷1930, Англия).

Аксиомы

A*1. $(A \vee A) \to A$ – *закон тавтологии

A*2. $A \to (B \vee A)$ – *закон добавления

A*3. $(A \vee B) \to (B \vee A)$ – *закон перестановки

A*4. $(A \vee B) \to ((C \vee A) \to (C \vee B))$ – *закон суммирования

Правила вывода

P1: Подстановка A вместо B;

P2: Замена на эквивалентную формулу $A \to B \equiv \overline{A} \vee B$

P3: **Modus ponens** $(A \to B, A) \vdash$ B.

Система аксиом современной исчисления высказываний состоит из 11 аксиом (по сути представляющих собой тождественно истинные формулы алгебры логики- тавтологии), которые делятся на четыре группы.

Первая группа аксиом (содержащая только импликацию).

I_1: $x \to (y \to x)$.

I_2: $(x \to (y \to z)) \to ((x \to y) \to (x \to z))$.

Вторая группа аксиом (к импликации присоединилась конъюнкция):

II_1: $x \wedge y \to x$.

II_2: $x \wedge y \to y$.

II_3: $(z \to x) \to ((z \to y) \to (z \to x \wedge y))$.

Третья группа аксиом (к импликации присоединилась дизъюнкция):

$\text{III}_1: x \to x \vee y$

$\text{III}_2: y \to x \vee y.$

$\text{III}_3: (x \to z) \to ((y \to z) \to (x \vee y \to z)).$

Четвертая группа аксиом (к импликации присоединилось отрицание):

$\text{IV}_1: (x \to y) \to (\bar{y} \to \bar{x})$

$\text{IV}_2: x \to \overset{=}{x}$

$\text{IV}_3: \overset{=}{x} \to x.$

Таким образом, множество аксиом исчисления высказываний, заданное 4 группами аксиом, бесконечно.

✓ **Замечание** В приведенной формальной аксиоматической системе логические знаки $\bar{},\wedge,\vee,\to$ воспринимаются как символы, а не как логические операции.

4.1 Правила вывода

Правила вывода определяют переход от посылок к следствиям, указывают, каким образом высказывания, истинность которых известна, могут быть видоизменены, чтобы получить новые истинные высказывания.

Основных правил вывода в исчислении высказываний два: *правило подстановки и правило простого заключения.*

4.2 Правило подстановки(ПП)

Если формула А выводима (доказуема) в исчислении высказываний, х- переменная, В- произвольная формула исчисления высказываний, то формула , полученная в результате замены в формуле А переменной х всюду , где она входит, формулой В, является также выводимой(доказуемой) формулой.

Операция замены в формуле А переменной х формулой В, носит название **подстановки** и символически записывается так

$$\int_X^B (A) \ .$$

Уточним сформулированное правило:

а) если формула А есть переменная х , то подстановка $\int_X^B (A)$ дает , очевидно, В;

б) если формула А есть переменная у , отличная от х ,то подстановка $\int_X^B (A)$ дает А;

в) подстановка формулы В вместо х в отрицание формулы А есть отрицание подстановки, т. е. подстановка $\int_X^B (\bar{A})$ дает $\overline{\int_X^B (A)}$;

г) если A_1 и A_2- некоторые формулы, то подстановка $\int_X^B (A_1 * A_2)$ дает $\int_X^B (A_1) * \int_X^B (A_2)$, где через символ * обозначен любой из символов операций конъюнкция, дизъюнкция или отрицание $\left(\wedge, \vee, ., \bar{}\right)$

Если А- выводимая (доказуемая) формула, то будем писать, $\vdash$**А. Читается «А доказуема».**Тогда правило подстановки (ПП) можно записать схематически следующим образом:

$$\frac{\vdash A}{\vdash \int_X^B (A)} \ .$$

И читается эта запись так: “Если формула А выводима (доказуема), то выводима (доказуема) и формула $\int_X^B (A)$.

4.3 Правило заключения (ПЗ)

Если формулы А и А→В выводимы (доказуемы) в исчислении высказываний, то формула В также выводима (доказуема). Схематическая запись этого правила имеет вид:

$$\frac{\vdash A;\ \vdash A \to B}{\vdash B} \qquad \textbf{(Modus ponens)}$$

«если верно, что из *A* следует *B* и *A* является истинным, то истинно *B*.»

Правомерность этого правила очевидна: если импликация и посылка истинны, то заключение в импликации может быть только истинным (см. таблицу истинности операции "импликация").

Пример: (*Modus ponens*)

Если лекция скучная, то студент спит. Лекция скучная.

Значит: студент спит

Множество правил вывода задано одной схемой, и также бесконечно.

1) **Modus tollens** (отрицательный марус)

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

Пример

если лекция скучная, то студент спит. Студент не спит

Значит : лекция не скучная

4.4 Определение выводимой (доказуемой) формулы

а) Всякая аксиома является доказуемой формулой.

б)Формула, полученная из доказуемой формулы путем применения подстановки вместо переменной х произвольной формулы В, есть доказуемая формула.

в) Формула В, полученная из доказуемых формул А и $A \to B$ путем применения ПЗ, есть доказуемая формула.

г) Никакая другая формула исчисления высказываний не считается доказуемой .

Процесс получения доказуемых формул будем называть ***доказательством (выводом) формул.*** Это процесс последовательного перехода от одной доказуемой формулы к другой с помощью аксиом, правила подстановки и правила заключения на каждом шаге (в определенном смысле это аналог равносильным преобразованиям в алгебре логики), так что вывод даже простой формулы может оказаться, в силу его многошаговости, достаточно громоздким.

4.5 Производные правила вывода

Производные правила вывода, как и рассмотренные правила подстановки и заключения, позволяют получать новые доказуемые формулы. Они получаются с помощью правил подстановки и заключения, а поэтому являются производными от них.

4.6 Правило сложной подстановки (СПП)

Пусть А – доказуемая формула; $x_1,\ldots,x_n$ - переменные, а $B_1,\ldots,B_n$ - любые формулы ИВ. Тогда результат одновременной подстановки в формулу А вместо $x_1,\ldots,x_n$ соответственно формул $B_1,\ldots,B_n$ является доказуемой формулой.

Схематично операция СПП записывается так:

$$\frac{\vdash A}{\vdash \int_{X_1,\ldots X_n}^{B_1,\ldots,B_n}(A)}$$

Так, в рассмотренном выше примере вместо шагов 4-5-6 и 9-10-11 можно было сразу применить СПП и тогда вместо 12 получим желаемый результат за 8 ходов:

1) $(x\to z)\to((y\to z)\to(x\vee y\to z))$ . . . .(Ш $_3$)

2) $(x\to y\vee x)\to((y\to y\vee x)\to(x\vee y\to y\vee x))$.

3) $y \to x \vee y$ . . .(Ш$_2$)

4) $x \to y \vee x$ . . . $\int_{X,Y,Z}^{Z,X,Y}$

5) $(y \to y \vee x) \to (x \vee y \to y \vee x)$...

6) $x \to x \vee y.$ . . . (Ш$_1$)

7) $y \to y \vee x$... $\int_{Y,X,Z}^{Z,Y,X}$

8) $x \vee y \to y \vee x,$ (5), (7), ПЗ

4.7 Правило сложного заключения

Правило сложного заключения также допускает обобщение.

Второе производное правило, получаемое в результате такого обобщения, применяется к формулам вида

$A_1 \to (A_2 \to (A_3 \to (...(A_n \to L)...)))$

и формулируется так :

Если формулы $A_1, A_2, ..., A_n$ и $A_1 \to (A_2 \to (A_3 \to (..(A_n \to L)...)))$ доказуемы, то и формула L доказуема.

Правило сложного заключения схематично записывается так:

$$\frac{\vdash A_1,\ \vdash A_2,\ \ldots,\ \vdash A_n,\ \vdash A_1 \to (A_2 \to (A_3 \to (\ldots(A_n \to L)\ \ldots)))}{\vdash L}$$

Следующие правила знакомы по тождественно истинным формулам алгебры логики, носящим те же наименования.

4.8 Правило силлогизма

Если доказуемы формулы A→B и B→C, то доказуема формула A→C , т. е.

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

4.9 Правило контр позиции

Если доказуема формула A→B, то доказуема формула $\overline{B} \to \overline{A}$, т. е.

$$\frac{\vdash \mathbf{A} \to \mathbf{B}}{\vdash \overline{B} \to \overline{A}}$$

На примере этого правила покажем, как доказываются такие утверждения в исчислении высказываний.

Сделаем одновременную подстановку $\int_{X,Y}^{A,B}(IV_1)$, получим доказуемую формулу $\vdash$(A→B)→$\vdash$($\overline{B} \to \overline{A}$).

Но по условию доказуема формула $\vdash$A→B.

по правилу заключения имеем $\vdash \overline{B} \to \overline{A}$.

4.10 Правило снятия двойного отрицания

а) Если доказуема формула $A \to \overline{\overline{B}}$, то доказуема формула $A \to B$.

б) Если доказуема формула $\overline{\overline{A}} \to B$, то доказуема формула $A \to B$.

Схематичная запись : $\dfrac{\vdash \mathbf{A} \to \overline{\overline{B}}}{\vdash A \to B}$ и $\dfrac{\vdash \overline{\overline{A}} \to \mathbf{B}}{\vdash A \to B}$

4.11 Понятие выводимости формул из совокупности формул

Будем рассматривать конечную совокупность формул H={A_1,A_2,…,A_n}.

➢ **Определение Формулы, выводимой из совокупности гипотез H**

1)Всякая формула $A_i \in H$,является формулой, выводимой из H.

2) Всякая доказуемая формула выводима из H.

3) Если формулы С и С→В выводимы из совокупности Н, то формула В также выводима из Н.

Если некоторая формула В выводима из совокупности Н, то это записывают так: **Н ⊢В**.

Нетрудно видеть, что класс формул, выводимых из совокупности Н, совпадает с классом доказуемых формул в случае, когда совокупность Н содержит только доказуемые формулы, и в случае, когда Н пуста.

Если же совокупность формул Н содержит хотя бы одну не доказуемую формулу, то класс формул, выводимых из Н, шире класса доказуемых формул.

Пример. Доказать, что из совокупности формул Н={А ,В} выводима формула $A \wedge B$.

Так как $A \in H$ и $B \in H$, то по определению выводимой формулы

Н ⊢А,

Н ⊢В.

Возьмем аксиомы II_3 и I_1, и выполним подстановки $\int_{X,Y,Z}^{A,B,A}(\text{II}_3)$ и $\int_{X,Y}^{B,A}(\text{I}_1)$.

В результате получим доказуемые формулы, которые выводимы из Н по определению выводимой формулы, т. е.

Н ⊢(А→А)→((А→В)→(А→ $A \wedge B$)),

Н ⊢В→(А→В),

Так как формула А→А доказуема, то Н ⊢А→А.

По правилу заключения получаем: Н ⊢(А→В)→(А→ $A \wedge B$)).

По правилу заключения получаем: Н ⊢А→В.

По правилу заключения получаем: Н ⊢А→ $A \wedge B$.

И, наконец, получаем: Н ⊢ $A \wedge B$

При доказательстве выводимости формулы из совокупности формул можно пользоваться не только основным правилом заключения, но и правилом сложного заключения.

Свойства выводимости

1. Если $H \vdash A$ и $H \subset W$, то $W \vdash A$.
2. Если $A \in H$, то $H \vdash A$.
3. Если $\vdash A$ и H произвольная совокупность гипотез, то $H \vdash A$.
4. Всякая формула выводима из пустой совокупности гипотез, является доказуемой формул.

Правила выводимости из гипотез

1). $\dfrac{H \vdash A,\ \ H \vdash A \to B}{H \vdash B}$

Глава 5 Понятие вывода

➢ **Определение Выводом** из конечной совокупности формул H называется всякая конечная последовательность формул $B_1,B_2,\ldots,B_к$, всякий член которой удовлетворяет одному из следующих трех условий:

1) он является одной из формул совокупности H,

2) он является доказуемой формулой,

3) он получается по правилу заключения из двух любых предшествующих членов последовательности $B_1,B_2,\ldots,B_к$.

Как было показано в предыдущем примере, выводом из совокупности формул H={A,B} является конечная последовательность формул:

A, B, $(A\to A)\to((A\to B)\to(A\to A\wedge B))$, $(A\to B)$, $A\to A$, $(A\to B)\to(A\to A\wedge B))$, $A\to B$, $A\to A\wedge B$, $A\wedge B$.

Если же здесь воспользоваться правилом сложного заключения , то вывод можно записать так:

A, B, $(A\to A)\to((A\to B)\to(A\to A\wedge B))$, $B\to(A\to B)$, $A\to A$, $A\to B$, $A\wedge B$.

Из определения выводимой формулы и вывода из совокупности формул следуют очевидные свойства вывода:

5.1 Свойства вывода

1)Всякий начальный отрезок вывода из совокупности H есть вывод из H.

2)Если между двумя соседними членами вывода из H (или в начале или в конце его) вставить некоторый вывод из H, то полученная новая последовательность формул будет также выводом из H.

3)Всякий член вывода из совокупности H является формулой, выводимой из H.

Всякий вывод из H является выводом его последней формулы.

4)Если $H\subset W$ (включено), то всякий вывод из H является выводом из W.

5)Для того, чтобы формула B была выводима из совокупности H, необходимо и достаточно, чтобы существовал вывод этой формулы из H.

5.2 Правила выводимости

Эти правила непосредственно следуют из свойств вывода с использованием ПП и ПЗ.

Пусть H и W – две совокупности формул исчисления высказываний. Будем обозначать через H, W их объединение, т. е. H,W= $H \cup W$.

В частности, если совокупность W состоит из одной формулы C, то будем записывать объединение $H \cup \{C\}$ в виде H,C.

5.3 Основные правила выводимости

1. $\underline{H \vdash A}$

Это правило следует непосредственно из определения вывода H,W ⊢A из совокупности формул :

"Если A выводима из H, то она выводима из $H \cup W$ ".

2. $\dfrac{H,C \vdash A, H \vdash C}{H \vdash A}$.

3. $\dfrac{H,C \vdash A,\ W \vdash C}{H,W \vdash A}$

4. $\dfrac{H \vdash C \to A}{H,C \vdash A}$

5. **Теорема дедукции**: Пусть H - множество формул, C, A – формулы, тогда $\dfrac{H, C \vdash A}{H \vdash C \to A}$.

В частности, если $H = \varnothing$, то если C ⊢ A $\Rightarrow$ C→A

Теорема о дедукции показывает, что для установления импликации $\Gamma \vdash A \to B$ достаточно показать $\Gamma, A \vdash B$, что часто бывает гораздо проще. В математической практике этому соответствует следующий пример рассуждения. Если нужно в некоторой ситуации установить, что $A \to B$, то ***допустим***

(***введем гипотезу***), что ***A*** верно, и докажем ***B***, исходя из этой гипотезы.

5A. **Обобщенная теорема дедукции**:

$$\frac{\{C_1, C_1, \ldots, C_k\} \vdash A}{\vdash C_1 \to (C_2 \to (C_3 \to \ldots (C_k \to A)\ldots))}$$

Теорема. (обратная теорема дедукции)

$H \vdash A \to B \Rightarrow H,\ A \vdash B$.

6. **Правило введения конъюнкции**: $\dfrac{H \vdash A, H \vdash B}{H \vdash A \wedge B}$

7. **Правило введения дизъюнкции**:

$$\frac{H, A \vdash C; H, B \vdash C}{H, A \vee B \vdash C}.$$

Примеры на использование теоремы дедукции

- **Пример** $\vdash (x \to (y \to z)) \to (y \to (x \to z))$

Рассмотрим последовательность формул:

$x \to (y \to z), y, x \vdash z$

$x \to (y \to z), y, x, y \to z, z$

- **Пример** $\vdash (x \to (y \to z)) \to (xy \to z)$

$x \to (y \to z), xy \vdash z$

$x \to (y \to z), xy, xy \to x, y \to z$

$xy \to y, y, z$

- **Пример** $\vdash (xy \to z) \to (x \to (y \to z))$

$xy \to z, x, y \vdash z$

$\vdash xy \to z$ (по условию)

$\vdash x$ (по условию)

$\vdash xy$ (по условию)

$\vdash z$

5.4 Построение вывода в логике высказываний

- **Пример** Докажем, что выводима формула $\vdash (\neg B \to \neg A) \to (A \to B)$.

Док-во

По теореме, обратной теореме дедукции, посылку можно перенести в левую часть:

$\neg B \to \neg A \vdash A \to B$.

Проделаем эту операцию еще раз:

$\neg B \to \neg A,\ A \vdash B$.

1. $\neg B \to \neg A$ – гипотеза.
2. A – гипотеза.

Формулу B удобно получить из аксиомы А3.

Поэтому запишем эту аксиому:

$(\neg B \to \neg A) \to ((\neg B \to A) \to B)$

К формулам 1 и 3 применим правило вывода Modus ponens

4. $(\neg B \to A) \to B$. MP 1, 3.

Посылку в формуле 4 можно получить из аксиомы А1, если заменить B на $\neg B$:

5. $A \to (\neg B \to A)$. А1 с подстановкой вместо B – $\neg B$.

Далее дважды применяем правило Modus ponens:

6. $\neg B \to A$. MP 2, 5.
7. B. MP 6, 4.

Вывод построен, и применением теоремы дедукции мы доказали выводимость первоначальной формулы.

Отметим, что вывод может быть неединственным, в частности, формулы могут быть записаны в другом порядке.

5.5 Доказательство некоторых законов логики

Правила выводимости, и особенно теорема дедукции, позволяют доказать ряд законов логики.

1. Закон перестановки посылок.

$\vdash(x \to (y \to z)) \to (y \to (x \to z))$. (1)

Доказательство:

Можно показать, что из совокупности формул H={x→(y→ z), y, x} следует вывод x→(y→ z), y, x, y→ z, z, т. е. из совокупности H выводима формула z.

Тогда по обобщенной теореме дедукции доказуема формула (1). И тогда по ПЗ из закона перестановки посылок вытекает правило перестановки посылок в доказуемых формулах: $\vdash(x \to (y \to z))$

$\vdash(y\to(x\to z))$. (2)

Действительно, если $\vdash x\to(y\to z)$, (2), то из (1) и (2) по правилу заключения следует $\vdash y\to(x\to z)$.

2. **Закон соединения посылок**

$\vdash(x\to(y\to z))\to(x\wedge y\to z)$. (3)

Доказательство:

Можно показать, что из совокупности формул $H=\{x\to(y\to z),\ x\wedge y\}$ следует вывод $x\to(y\to z)$, $x\wedge y$, $x\wedge y\to x$, $x\wedge y\to y$, x, y, $y\to z$, z, т. е. из совокупности H выводима формула z. Тогда по обобщенной теореме дедукции доказуема формула (3).

Из закона соединения посылок вытекает правило соединения посылок в доказуемых формулах: $\vdash(x\to(y\to z))$

$\vdash x\wedge y\to z$. (4)

Действительно, если $\vdash x\to(y\to z)$, (4), то из (3) и (4) по правилу заключения следует $\vdash x\wedge y\to z$. (5)

Закон разъединения посылок

Так как из совокупности формул $H=\{x, y, x\wedge y\to z\}$ следует вывод x, y, $x\wedge y\to z$, $x\wedge y$, z, то из совокупности формул H выводима формула z. Тогда по обобщенной теореме дедукции доказуема формула (5).

Из закона разъединения посылок вытекает правило разъединения посылок в доказуемых формулах: $\vdash x\wedge y\to z$

$\vdash x\to(y\to z)$. (6)

Действительно, если $\vdash x\wedge y\to z)$, (6), то из (5) и (6) по правилу заключения следует

$\vdash x\to(y\to z)$.

3. $\vdash x\to(\bar{x}\to y)$. (7)

Доказательство:

Сделаем подстановки в аксиомы I_1 и IV_1: $\int_{Y}^{\bar{Y}}(I_1)$ и $\int_{X,Y}^{\bar{Y},X}(IV_1)$.

В результате получим доказуемые формулы:

$\vdash x \to (\bar{y} \to x)$, (8) и $\vdash (\overline{y} \to x) \to (\bar{x} \to \bar{\bar{y}})$. (9)

Из формул (8)и (9)по правилу силлогизма следует: $\vdash x \to (\bar{x} \to \bar{\bar{y}})$.

Используя закон соединения посылок, получим: $\vdash x \wedge \bar{x} \to \bar{\bar{y}}$.

Используя правило снятия двойного отрицания, получим: $\vdash x \wedge \bar{x} \to y$.

И, наконец, применяя закон разъединения посылок, получим (7).

5. Закон исключенного третьего $\vdash x \vee \bar{x}$.

Доказательство

Воспользуемся доказуемой формулой $\vdash \overline{x \vee y} \to \bar{x} \wedge \bar{y}$ (10) и, сделав в ней подстановку $\int_{Y}^{\bar{X}}(10)$, получим :

$\vdash \overline{x \vee \bar{x}} \to \bar{x} \wedge \bar{\bar{x}}$. (11)

Также сделаем подстановку в формуле (7), заменяя x на $\bar{x}$, а у на $\bar{y}$:

$\vdash \bar{x} \to (\bar{\bar{x}} \to \bar{y})$. (12)

Используя закон соединения посылок, будем иметь: $\vdash \bar{x} \wedge \bar{\bar{x}} \to \bar{y}$. (13)

Из формул (11) и (13) по правилу силлогизма получаем $\vdash \overline{x \vee \bar{x}} \to \bar{y}$. (14)

Из формулы (14) по правилу контрапозиции следует $\vdash \bar{\bar{y}} \to \overline{\overline{x \vee \bar{x}}}$.

Используя оба правила снятия двойного отрицания, получаем $\vdash y \to x \vee \bar{x}$ (15)

Пусть теперь у- любая доказуемая формула R, тогда из формул $\vdash R$, $\vdash R \to x \vee \bar{x}$ по правилу заключения получаем $\vdash x \vee \bar{x}$.

6. $\vdash \bar{x} \wedge \bar{y} \to \overline{x \vee y}$.

Примем этот закон без доказательства.

Задачи для самостоятельного решения

1. Являются ли выводами в ИВ следующие последовательности формул:

а) $(A \to (A \vee e))$;

б) $(A \to (A \vee e))$,

$((A \to (A \vee e)) \to (e \to (A \to (A \vee e))))$, $(e \to (A \to (A \vee e)))$;

в) $(A \to (e \to A))$, $((A \to (e \to A)) \to e)$, B?

2. Вывести в ИВ формулы:

а) $((A \vee A) \to A)$;

б) $(A \to \neg\neg A)$.

3. Являются ли выводами в ИП следующие последовательности:

а) $(\forall x \forall y U(x,y) \to \exists y U(x,y))$;

б) $(\forall x P(x) \to P(y))$, $(\forall x P(x) \to \forall y P(y))$, где P - одноместный предикатный символ;

в) $(U(x) \to \exists x U(x))$,

$((U(x) \to \exists x U(x)) \to (\forall x U(x) \to (U(x) \to \exists x U(x))))$,

$(\forall x U(x) \to (U(x) \to \exists x U(x)))$?

4. Каким требованиям должна удовлетворять формула $U(x)$, чтобы следующая последовательность была выводом в ИП:

а) $(U(y) \to \exists x U(x))$, $(\exists y U(y) \to \exists x U(x))$;

б) $(\forall x U(x) \to U(y))$, $(\forall x U(x) \to \forall y U(y))$?

5. Построить выводы формул:

а) $(\forall x \forall y U(x,y) \to \forall y \forall x U(x,y))$;

б) $(\exists x \exists y U(x,y) \to \exists y \exists x U(x,y))$;

в) $(\exists x \forall y U(x,y) \to \forall y \exists x U(x,y))$.

3.5. Являются ли следующие последовательности формул выводами из $\Gamma = \{(C \to B(x))\}$, где C не содержит свободных вхождений x:

а) $(C \to B(x))$, $(C \to \forall x B(x))$;

б) $((C \to B(x)) \to (M(y) \to (C \to B(x))))$, $(C \to B(x))$, $(M(y) \to (C \to B(x)))$, $(\exists y M(y) \to (C \to B(x)))$, если C и $B(x)$ не содержат свободных вхождений y?

3.6. Построить выводы из $\Gamma = \{\forall x(C(x) \to B(x))\}$ следующих формул:

а) $(\exists x C(x) \to \exists x B(x))$;

б) $(\forall y C(y) \to \forall z B(z))$.

Глава 6 Связь между АВ и ИВ

Формулы исчисления высказываний можно интерпретировать как формулы алгебры высказываний. Для этого будем трактовать переменные исчисления высказываний как переменные алгебры высказываний, т. е. переменные в содержательном смысле, принимающие два значения: истина и ложь (1 и 0).

Операции $\vee,\wedge,\rightarrow u -$ определим так же, как в алгебре высказываний.

При этом всякая формула исчисления высказываний при любых входящих в нее переменных будет принимать одно из значений 1 или 0, вычисляемое по правилам алгебры высказываний.

Введем понятие значения формулы исчисления высказываний. Пусть А- формула исчисления высказываний, $x_1,x_2,\ldots,x_n$- попарно различные переменные, среди которых находятся все переменные, входящие в формулу А. Обозначим через $a_1, a_2,\ldots,a_n$ набор значений этих переменных, состоящих из 1 и 0, длины n. Очевидно, что вектор $(a_1, a_2,\ldots,a_n)$ имеет 2^n значений.

Имеют место три теоремы, которые устанавливают связь между основными фактами алгебры высказываний и исчисления высказываний.

❖ **Теорема 1** Каждая формула, доказуемая в исчислении высказываний, является тождественно истинной в алгебре высказываний.

Формулировка этой теоремы содержит в себе три положения:

1)Каждая аксиома исчисления высказываний – тождественно истинная формула в алгебре высказываний.

2)Правило подстановки, примененное к тождественно истинным формулам, приводит к тождественно истинным формулам.

3)Правило заключения, примененное к тождественно истинным формулам, приводит к тождественно истинным формулам.

❖ **Теорема 2(о выводимости)** Пусть A –некоторая формула исчисления высказываний; $x_1,x_2,\ldots,x_n$ – набор переменных, содержащих все переменные, входящие в формулу A; $a_1, a_2,\ldots,a_n$ – произвольный фиксированный набор значений этих переменных. Обозначим через H конечную совокупность формул

$$H = \left\{x_1^{a1}, x_2^{a2}, \ldots, x_n^{an}\right\}, \text{ где } x_i^{ai} = \begin{cases} x_i, если\_a_i = 1, \\ \overline{x_i}, если\_a_i = 0. \end{cases}$$

Тогда: Если $R_{a1,a2,..,an}(A)=1$, то $H \vdash A$.

1) Если $R_{a1,a2,..,an}(A)=0$, то $H \vdash \overline{A}$, где $R_{a1,a2,..,an}(A)$– значение формулы A на наборе $a_1, a_2,\ldots,a_n$.

❖ **Теорема 3** Каждая тождественно истинная формула алгебры высказываний доказуема в исчислении высказываний.

6.1 Правила подстановки и замены

Справедливы правило подстановки и правило замены.

Пусть F и G – формулы, содержащие букву A, F_H^A и G_H^A – формулы, полученные из формул F и G соответственно подстановкой вместо буквы A формулы H.

Правило подстановки. Если формула F логически эквивалентна формуле G, то формула F_H^A логически эквивалентна формуле G_H^A.

Пусть F_G – формула, в которой выделена некоторая подформула G, F_H – формула, полученная из формулы F_G заменой G на некоторую формулу H.

Правило замены. Если формулы G и H логически эквивалентны, то логически эквивалентны и формулы F_G и F_H.

Доказательства правил подстановки и замены основано на сравнении таблиц истинности соответствующих формул.

➧ **Пример** $(A \to B) \leftrightarrow (\neg A \vee B)$

Док-во: По правилу подстановки, $(\neg A \to B)$ эквивалентна формуле $(\neg\neg A \vee B)$.

По правилу замены, $(\neg\neg A \vee B)$ эквивалентна формуле $(A \vee B)$.

Следовательно, по свойству транзитивности, формулы $(\neg A \to B)$ и $(A \vee B)$ логически эквивалентны.

➢ **Определение**. Говорят, что формула A *логически влечет* формулу B, если формула $A \to B$ является тавтологией.

❖ **Теорема**. Отношение логического следствия - отношение предпорядка, то есть рефлексивно и транзитивно.

6.2Проблемы аксиоматического исчисления высказываний

Всякая аксиоматическая теория для ее обоснования требует рассмотрения четырех проблем:

1) проблемы разрешимости,
2) проблемы непротиворечивости,
3) проблемы полноты,
4) проблемы независимости.

6.3 Проблема разрешимости исчисления высказываний

Проблема разрешимости исчисления высказываний заключается в доказательстве существования алгоритма, который позволил бы для любой заданной формулы исчисления высказываний определить, является ли она доказуемой или не является.

Имеет место теорема:

❖ **Теорема** Проблема разрешимости для исчисления высказываний разрешима.

Действительно, любая формула исчисления высказываний может рассматриваться как формула алгебры высказываний, и, следовательно, можно рассматривать ее логические значения на различных наборах значений входящих в нее переменных.

6.4 Проблема полноты исчисление высказываний

➢ **Определение Аксиоматическое исчисление высказываний называется полным в узком смысле**, если добавление к списку его аксиом любой недоказуемой в исчислении формулы в качестве новой аксиомы приводит к противоречивому исчислению.

➢ **Определение** Исчисление высказываний называется **полным в широком смысле**, если любая тождественно истинная формула в нем доказуема.

Из этих определений следует, что проблема полноты исчисления высказываний содержит два вопроса:

1) Можно ли расширить систему аксиом аксиоматического исчисления путем добавления к ней в качестве новой аксиомы какой-нибудь недоказуемой в этом исчислении формулы?

2) Является ли всякая тождественно истинная формула алгебры высказываний доказуемой в исчислении высказываний?

Рассмотренное нами исчисление высказываний полно как в узком смысле, так и в широком.

6.5 Проблема независимости аксиом исчисления высказываний

Для всякого аксиоматического исчисления возникает вопрос о независимости его аксиом. Вопрос этот ставится так: можно ли какую-нибудь аксиому вывести из остальных аксиом, применяя правила вывода данной системы?

Если для некоторой аксиомы системы это возможно, то эту аксиому можно исключить из списка аксиом системы, и

логическое исчисление при этом не изменится, т. е. класс доказуемых формул останется без изменений.

➢ **Определение** Аксиома А называется **независимой** от всех остальных аксиом исчисления, если она не может быть выведена из остальных аксиом.

➢ **Определение** Система аксиом исчисления называется **независимой**, если каждая аксиома системы независима.

Рассмотренная нами система аксиом исчисления высказываний независима.

Глава 7 Автоматическое доказательство теорем

Пусть имеется множество формул $\Gamma = \{A_1, A_2, \ldots, A_n\}$ и формула B.

➢ **Определение** *Автоматическим доказательством формулы B* называют алгоритм, который проверяет вывод

$$A_1, A_2, \ldots, A_n \vdash B.$$

Если посылки A_1, A_2, ..., A_n истинны, то истинно заключение B. Или

Если причины A_1, A_2, ..., A_n имели место, то будет иметь место следствие B.

Проблема доказательства в логике состоит в том, чтобы установить, что если истинны формулы $A_1, A_2, \ldots, A_n$, то истинна формула B.

В общем случае такой алгоритм построить **нельзя**. Но для некоторых частных случаев такие алгоритмы существуют.

Доказательство теоремы равносильно доказательству общезначимости некоторой формулы. Наиболее эффективно доказательство общезначимости формул осуществляется методом резолюций.

Процедура поиска доказательства методом резолюций фактически является процедурой поиска опровержения, т. е. вместо доказательства общезначимости формулы доказывается, что отрицание формулы противоречиво: $A \equiv И \Leftrightarrow \neg A \equiv Л$.

Робинсон пришел к заключению, что правила вывода, которые следует применять при автоматизации процесса доказательства с помощью компьютера, не обязательно должны совпадать с правилами вывода, используемыми человеком. Он обнаружил, что общепринятые правила вывода, например, **правило modus ponens**, специально сделаны "слабыми", чтобы человек мог интуитивно проследить за каждым шагом процедуры доказательства.

Правило резолюции более сильное, оно трудно поддается восприятию человеком, но эффективно реализуется на компьютере.

7.1 Метод резолюций

Метод предложен Дж. Робинсоном в 1965 году и по сей день лежит в основе большинства систем поиска логического вывода.

Метод резолюций был использован в качестве основы нового языка программирования. Так в 1972 году родился язык Пролог ("ПРОграммирование в терминах ЛОГики"), быстро завоевавший популярность во всем мире.

➢ **Определение Метод резолюций** - аксиоматическая теория первого порядка, которая использует доказательство от противного, и, не использует аксиоматику исчисления предикатов.

1. Язык метода резолюции - *язык дизъюнктов.*
2. Аксиомы только *собственные.*
3. Правило вывода - *резолюция*

➢ **Определение Литера -** выражения вида A или $\neg A$.

➢ **Определение**. **Предложение-** дизъюнкция формул вида A или $\neg A$

➢ **Определение** Литеры A и $\neg A$ называются *контрарными*, а множество $\{A, \neg A\}$ – *контрарной парой.*

➢ **Определение Дизъюнкт** – это дизъюнкция литер (или элементарная дизъюнкция).

Пример

$A \vee B \vee C$ – дизъюнкт; $A \vee \neg B$ – дизъюнкт; $A \vee B \,\&\, C$ – не дизъюнкт;

$\neg A$ – дизъюнкт.

➢ **Определение** Дизъюнкт называется *пустым*, (обозначается), если он не содержит литер.

Пустой дизъюнкт всегда ложен, так как в нем нет литер, которые могли бы быть истинными при любых наборах переменных.

Правило вывода резолюция использует расширенный принцип силлогизма и унификацию.

Традиционный силлогизм: A → B, B → C □_ A → C

Применительно к дизъюнктивной записи можно представить как

*A ∨ B *A ∨ B ∨ *D

*B ∨ C или "обобщенный" вариант *B ∨ C ∨ E

*A ∨ C *A ∨ C ∨ *D ∨ E

Унификация позволяет заменить переменную x на терм t. То есть вместо переменной могут быть подставлены константа или другая переменная (из той же области), или функция, область значений которой совпадает с областью определения x.

a(const)→x←y(из той же области)

↑

f(z)

Вывод здесь заключается в том, что в систему добавляется отрицание формулы (дизъюнкта!), которую необходимо вывести. Вывод состоит в последовательном применении резолюции до получения пустого дизъюнкта.

Пример Можно сказать, что это прообраз или предельно упрощенный вариант «системы искусственного интеллекта».

Пусть мир описывается двумя аксиомами:

Миша повсюду ходит за Леной: A1.∀x(B(Л, x) → B(M, x))

Лена в университете:A2.B(Л, У)

Требуется доказать (ответить на вопрос)

Где Миша? A3. ∃ x B(M, x) ?

Решение

Вопрос (доказываемую формулу с добавленным знаком вопроса) ∃ x B(M, x)? преобразуем в .* ∃ x B(M, x) (отрицание вопроса). Далее задвигаем отрицание за квантор, производим сколемизацию и добавляем специальный «предикат ответа», который будет аккумулировать процесс унификации).

В результате получаем дизъюнкт:

.* B(M, x) ∨ Отв(M, x)

Вся система (две аксиомы и вопрос) будет состоять из трех дизъюнктов:

Д1: *B(Л, x) ∨ B(M, x)

Д2:B(Л, У)

Д3: *B(M, x) ∨ Отв(M, x)

Вывод:

Резолюция Д1-Д2 дает Д4: B(M,У)

Резолюция Д4-Д3 дает Д5: ∨ **Ответ(M,У)** можно интерпретировать как «Миша в университете».

Метод резолюций – это метод автоматического доказательства теорем. Это алгоритм, проверяющий отношение выводимости $\Gamma \vdash A$.

В общем случае алгоритм автоматического доказательства теорем не существует, но для формальных теорий (таких как исчисление высказываний, исчисление предикатов) подобные алгоритмы известны.

Любая формула исчисления высказываний может быть преобразована в предложение следующей последовательностью действий:

Замена импликации по формуле: $A \to B = \neg A \vee B$ (. В результате в формуле остаются связки: $\neg$, $\vee$, $\wedge$).

Преобразование выражений с $\neg\neg A = A$, законам де Моргана: $\neg(A \vee B) = \neg A \wedge \neg B$, $\neg(A \wedge B) = \neg A \vee \neg B$. В результате инверсии остаются только перед буквами.

Приведение формулы к КНФ с помощью дистрибутивных законов:

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C),$$
$$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

❖ **Теорема** *Правилом резолюции* называют следующее правило вывода:

$$\frac{A \vee B,\ \neg A \vee C}{B \vee C}.$$

Правило можно также записать в следующем виде:

$$A \vee B, \neg A \vee C \vdash B \vee C.$$

Доказательство

Правило резолюций можно доказать, используя равносильности логики высказываний:

$(A \vee B)$ & $(\neg A \vee C) \supset (B \vee C) = \neg((A \vee B)$ & $(\neg A \vee C)) \vee (B \vee C) = \neg(A \vee B) \vee \neg(\neg A \vee C) \vee (B \vee C) = \neg A \& \neg B \vee A \& \neg C \vee (B \vee C) = (\neg A \vee A)$ & $(\neg A \vee \neg C)$ & $(\neg B \vee A)$ & $(\neg B \vee \neg C) \vee (B \vee C) = (\neg A \vee \neg C)$ & $(\neg B \vee A)$ & $(\neg B \vee \neg C) \vee (B \vee C) = (\neg A \vee \neg C \vee B \vee C)$ & $(\neg B \vee A \vee B \vee C)$ & $(\neg B \vee \neg C \vee B \vee C) =$ И.

Итак, при истинных посылках истинно заключение.

Дизъюнкт $B \vee C$ называется *резольвентой* дизъюнктов $A \vee B$ и $\neg A \vee C$ по литере A:

$$B \vee C = res_A(A \vee B \text{ и } \neg A \vee C).$$

– единственное правило, применяемое в методе резолюций, что позволяет не запоминать многочисленных аксиом и правил вывода.

Если дизъюнкты не содержат контрарных литер, то резольвенты у них не существует.

Для дизъюнктов A и $\neg A$ резольвента есть пустой дизъюнкт:

$$res_A(A, \neg A) = \ .$$

Пример.

Пусть $F = A \vee B \vee C$, $G = \neg A \vee \neg B \vee D$.

Тогда

$res_A(F, G) = B \vee C \vee \neg B \vee D$.

$res_B(F, G) = A \vee C \vee \neg A \vee D$.

$res_C(F, G)$ не существует.

Метод резолюций соответствует методу доказательства от противного.

Действительно, условие $A_1, A_2, \ldots, A \vdash B$ равносильно условию $A_1, A_2, \ldots, A_n, \neg B \vdash \ .$

Метод резолюций относится к методам *непрямого вывода*.

Изложим процедуру вывода $A_1, A_2, \ldots, A_n \vdash B$ в виде алгоритма.

7.2 Алгоритм построения вывода методом резолюций

Шаг 1. Формулы $A_1, A_2, \ldots, A_n$ и формулу $\neg B$ привести к КНФ.

Шаг 2. Составить множество S дизъюнктов формул $A_1, A_2, \ldots, A_n$ и $\neg B$.

Шаг 3. Вместо пары дизъюнктов, содержащих контрарные литеры записать их резольвенту по правилу (2).

Шаг 4. Процесс продолжаем. Если он заканчивается пустым дизъюнктом, то вывод обоснован.

Изложенный алгоритм называется *резолютивным выводом из S.*

Возможны три случая:

1. Среди множества дизъюнктов нет содержащих контрарные литеры. Это означает, что формула B не выводима из множества формул $A_1, A_2, \ldots, A_n$.

2. В результате очередного применения правила резолюции получен пустой дизъюнкт. Это означает, что что формула B выводима из множества формул $A_1, A_2, \ldots, A_n$.

3. Процесс зацикливается, т. е. получаются все новые и новые резольвенты, среди которых нет пустых. Это ничего не означает.

Пример $A \supset (B \supset C), A\&B \vdash C$.

Применим для этого примера метод резолюций. Для этого нужно проверить вывод

$A \supset (B \supset C), A\&B, \neg C \vdash$.

Будем действовать в соответствии с алгоритмом.

Шаг 1. Нужно привести к КНФ формулы $A \supset (B \supset C), A\&B, \neg C$.

$A \supset (B \supset C) \equiv \neg A \vee (B \supset C) \equiv \neg A \vee \neg B \vee C$.

Формулы $A\&B, \neg C$ уже находятся в КНФ.

Шаг 2. Составим множество S дизъюнктов:

$S = \{\neg A \vee \neg B \vee C, A, B, \neg C\}$.

Шаг 3. Построим резолютивный вывод из S. Для этого выпишем по порядку все дизъюнкты из S:

$\neg A \vee \neg B \vee C$;

A;

B;

$\neg C$;

Вместо пары дизъюнктов, содержащих контрарные литеры запишем их резольвенту (в скобках указаны номера формул, образующих резольвенту):

$\neg B \vee C$.	(1, 2)
C.	(3, 5)
.	(4, 6)

Вывод заканчивается пустым дизъюнктом, что является обоснованием вывода $A \supset (B \supset C), A\&B \vdash C$.

Пример Записать с помощью формул логики высказываний и решить методом резолюций следующую задачу:

«Чтобы хорошо учиться, надо прикладывать усилия. Тот, кто хорошо учится, получает стипендию. В данный момент студент прикладывает усилия. Будет ли он получать стипендию?

Решение

Введем следующие высказывания:

A = "студент хорошо учится".

B = "студент прикладывает усилия".

C = "студент получает стипендию"

Чтобы утвердительно ответить на вопрос задачи: "Будет ли студент получать стипендию?", нужно проверить вывод:

$B \supset A, A \supset C, B \vdash C$.

Будем действовать в соответствии с алгоритмом.

Шаг 1. Нужно привести к КНФ формулы $B \supset A, A \supset C, B, \neg C$.

$B \supset A = \neg B \vee A$,

$A \supset C = \neg A \vee C$,

Формулы B и $\neg C$ уже находятся в КНФ.

Шаг 2. Составим множество S дизъюнктов:

$S = \{\neg B \vee A, \neg A \vee C, B, \neg C\}$.

Шаг 3. Построим резолютивный вывод из S. Сначала перепищем по порядку дизъюнкты из S:

1) $\neg B \vee A$.

2) $\neg A \vee C$.

3) B.

4) $\neg C$.

Затем вместо пары дизъюнктов, содержащих контрарные литеры запишем их резольвенту:

5) $\neg B \vee C$. (1, 2)

C. (3, 5)

. (4, 6)

Таким образом, на вопрос задачи можно ответить утвердительно: "Студент будет получать стипендию".

Правило резолюций более общее, чем правило *modus ponens* и производные правила.

Правило модус поненс также можно считать частным случаем правила резолюции при ложном А.

Докажем методом резолюций правило *modus ponens*. Необходимо построить вывод

$A, A \supset B \vdash B$.

Построим резолютивный вывод.

$A, \neg A \vee B \vdash B$.

$A, \neg A \vee B, \neg B \vdash$.

$S = \{A, \neg A \vee B, \neg B\}$.

A.

$\neg A \vee B$.

$\neg B$.

B.

✓ **Замечание** Недостатком метода резолюций является необходимость представления формул в КНФ.

✓ **Замечание** Автоматическое доказательство теорем методом резолюций основан на переборе и этот перебор может быть настолько большим, что затраты времени на него практически неосуществимы. Эти обстоятельства стимулируют поиски различных модификаций метода резолюций.

Приведем еще один пример применения метода резолюций, основанного на попарном переборе дизъюнктов.

Пример Построим с помощью метода резолюций следующий вывод:

$\neg A \supset B, C \vee A, B \supset \neg C \vdash A$,

Или, что то же: $\neg A \supset B, C \vee A, B \supset \neg C, \neg A \vdash$.

Перепишем все посылки в виде дизъюнктов: $A \vee B$, $C \vee A$, $\neg B \vee \neg C, \neg A \vdash$.

Выпишем по порядку все посылки и начнем их по очереди склеивать по правилу резолюций:

$A \vee B$

$C \vee A$

$\neg B \vee \neg C$

$\neg A$

$A \vee \neg C$ (1, 3)

B (1, 4)

$A \vee \neg B$ (2, 3)

C.	(2, 4)
A	(2, 5)
$\neg C$	(3, 6)
$\neg B$	(3, 8)
$\neg C$	(4, 5)
$\neg B$	(4, 7)
	(4, 9)

Мы видим, что такая стратегия перебора неэффективна. В данном случае существует более быстрый вывод.

Например:

5)	B.	(1, 4)
6)	C.	(2, 4)
7)	$\neg B$.	(3, 6)
8)		(5, 7)

Глава 8 Теории первого порядка

Логика первого порядка (*исчисление предикатов*) — формальное исчисление, допускающее высказывания относительно переменных, фиксированных функций и предикатов. Расширяет логику высказываний, является частным случаем логики высшего порядка.

Слова «первого порядка» указывают на отличие рассматриваемых теорий от таких теорий, в которых либо допускаются предикаты, имеющие в качестве возможных значений своих аргументов другие предикаты и функции, либо допускаются кванторы по предикатам или кванторы по функциям.

Логика первого порядка дает возможность строго рассуждать об истинности и ложности утверждений и об их взаимосвязи, в частности, о логическом следовании одного утверждения из другого.

Логика высказываний оперирует простейшими высказываниями, которые могут быть или истинными, или ложными. В разговорном языке встречаются более сложные повествовательные предложения, истинность которых может меняться при изменении объектов, о которых идет речь. В логике такие предложения, истинность которых зависит от параметров, обозначают с помощью предикатов.

"Предикат" с английского переводится как сказуемое **(praedicatum - сказанное).** Формально предикатом называется функция, аргументами которой могут быть ПРОИЗВОЛЬНЫЕ ОБ'ЕКТЫ из некоторого множества, а значения функции "истина" или "ложь". Предикат можно рассматривать как расширение понятия высказывания.

➢ **Определение Логика предикатов** - раздел логических теорий, в котором изучаются общезначимые связи между высказываниями о свойствах и отношениях предметов; в основе лежит формализованный язык, отображающий субъективно-предикатную структуру высказываний.

Пример. Вместо трех высказываний

"Маша любит кашу" ,"Даша любит кашу","Саша любит кашу"можно написать один предикат "Икс любит кашу"и договориться, что вместо неизвестного Икс могут быть либо Маша, либо Даша, либо Саша.

Подстановка вместо Икс имени конкретного ребенка превращает предикат в обычное высказывание.

В алгебре логики высказывания рассматриваются как нераздельные целые и только с точки зрения их истинности или ложности. Ни структура высказываний, ни, тем более, их содержание не затрагиваются.

В связи с этим возникает необходимость в расширении логики высказываний, в построении такой логической системы, средствами которой можно было бы исследовать структуру тех высказываний, которые в рамках логики высказываний рассматриваются как элементарные.

Такой логической системой является логика предикатов, содержащая всю логику высказываний в качестве своей части.

➢ **Определение Предикат** - это высказывание-функция, значение (истина/ложь) которого зависит от параметров.

Пример Предикат $x \le 2$ при $x = 1$ истина при $x = 3$ ложь.

➢ **Определение Одноместный предикат** P(x) - произвольная функция переменного x, определенная на множестве M и принимающая значение из множества {1; 0}.

✓ **Замечание** Высказывания – это нульместный предикат, логическая (пропозициональная) переменная, принимающая значения из множества {1;0}-. нульместный предикат.

➢ **Определение** Множество M, на котором определен предикат P(x), называется **областью определения предиката P(x).**

➢ **Определение** Множество всех элементов $x \in M$, при которых предикат принимает значения “истина” (1), называется **множеством (областью) истинности предиката** P(x), т.е. множество истинности предиката P(x)- это множество

$$I_p = \{x : x \in M, P(x) = 1\},$$

или иначе: $\underset{x}{M}[P]$ или так: $\underset{x}{M}[P(x)]$.

➧ **Пример** предикат P(x) – "x – простое число" определен на множестве N, а множество истинности I_P для него есть множество всех простых чисел.

➧ **Пример** Предикат Q(x) – "sinx=0" определен на множестве R, а его множеством истинности является $I_Q = \{k\pi, k \in Z\}$.

Из приведенных примеров видим, что одноместные предикаты выражают свойства предметов (субъектов).

Но предикаты могут быть не только одноместные.

Это просто проиллюстрировать, если представить, что дети могут любить не только кашу... "Икс любит Игрека" - двухместный предикат. "ВСЕ любят Игрека" - одноместный предикат. "ВСЕ любят КОЙ-КОГО [некоторого]" - нульместный предикат, то есть высказывание.

➢ **Определение Двухместный предикат P(x,y)** - функция двух переменных x и y, определенная на множестве $M=M_1xM_2$ и принимающая значения из множества {1;0}.

➧ **Пример** Q(x, y) – "x=y" - предикат равенства на множестве $RxR=R^2$;

✓ **Замечание** Предикаты при подстановки переменных становятся высказываниями, поэтому с предикатами можно производить все логические операции.

➢ **Определение** Предикат P(x), определенный на множестве M, называется **тождественно истинным,** если его множество истинности совпадает с областью определения, т. е. $I_p=M$.

➢ **Определение** Предикат P(x), определенный на множестве M, называется **тождественно ложным,** если его множество истинности является пустым множеством, т. е. $I_p=0$.

✓ **Замечание** Предикат P называется *тождественно истинным* (*тождественно ложным*), если на всех наборах своих переменных принимает значение 1 (0), *выполнимым,* если на некотором наборе своих переменных принимает значение 1.

Пример $P(x) = x \le 2$, $\forall x \quad x \le 2$ ложь $P(x) = x \le 2$, $\exists x \quad x \le 2$ - истина.

Обобщением понятия одноместного предиката является понятие многоместного предиката, с помощью которого выражаются отношения между предметами.

➢ **Определение n-местный предикат** - это функция $P(x_1, x_2, \ldots x_n)$ определенная на наборах длины n элементов некоторого множества M, принимающая значения в области True, False. Множество M называется *предметной областью предиката*, а x1, x2, ..xn *–предметными переменными*.

$$P(x_1, x_2, \ldots x_n) \to \{u, л\}.$$

Пример n- местный предикат- алгебраическое уравнение с n неизвестными $R(x_1, x_2, \ldots, x_n)$: $a_1 x_1 + \ldots + a_n x_n = 0$.

Пример Булева функция - n -местный предикат.

8.1 Логические операции над предикатами

Предикаты так же, как высказывания, могут принимать два значения: “истина” (1) и “ложь” (0), поэтому к ним применимы все операции логики высказываний, в результате чего из элементарных предикатов формируются сложные предикаты (как и в логике высказываний, где из элементарных высказываний формировались сложные, составные).

Рассмотрим применение операций логики высказываний к предикатам на примерах одноместных предикатов. Эти операции в логике предикатов сохраняют тот же смысл, который был им присвоен в логике высказываний.

Пусть на некотором множестве M определены два предиката P(x) и Q(x).

➢ **Определение Отрицание** предиката P(x) - предикат $\overline{P}(x)$ или $\overline{P(x)}$, который принимает значение “истина” при всех значениях $x \in M$, при которых предикат P(x) принимает значение “ложь”, и принимает значение “ложь” при тех

значениях $x \in M$, при которых предикат P(x) принимает значение "истина".

Очевидно, что $I_{\overline{P}} = \overline{I_P}$, т.е. множество истинности предиката $\overline{P}(x)$ является дополнением к множеству I_P.

➢ **Определение Конъюнкция** двух предикатов P(x) и Q(x) - предикат $P(x) \wedge Q(x)$, который принимает значение "истина" при тех и только тех значениях $x \in M$, при которых каждый из предикатов принимает значение "истина", и принимает значение "ложь" во всех остальных случаях.

Очевидно, что областью истинности предиката $P(x) \wedge Q(x)$ является общая часть области истинности предикатов P(x) и Q(x), т.е. пересечение $I_P \cap I_Q$.

Пример, для предикатов P(x): "x – четное число" и Q(x): "x кратно 3" конъюнкцией $P(x) \wedge Q(x)$ является предикат "x – четное число и x кратно трем", т.е. предикат "x делится на 6".

➢ **Определение Дизъюнкция** двух предикатов P(x) и Q(x) - предикат $P(x) \vee Q(x)$, который принимает значение "ложь" при тех и только тех значениях $x \in M$, при которых каждый из предикатов принимает значение "ложь", и принимает значение "истина" во всех остальных случаях.

Ясно, что областью истинности предиката $P(x) \vee Q(x)$ является объединение области истинности предикатов P(x) и Q(x), т.е. $I_P \cup I_Q$.

➢ **Определение Импликация** предикатов P(x) и Q(x) - предикат $P(x) \to Q(x)$, который является ложным при тех и только тех значениях $x \in M$, при которых одновременно P(x) принимает значение "истина", а Q(x) – значение "ложь", и принимает значение "истина" во всех остальных случаях.

Поскольку при каждом фиксированном $x \in M$ справедлива равносильность

$$P(x) \to Q(x) \overset{18}{\equiv} \overline{P}(x) \vee Q(x), \text{ то } I_{P \to Q} = I_{\overline{P}} \cup I_Q.$$

➢ **Определение Эквиваленция** предикатов P(x) и Q(x) - предикат $P(x) \leftrightarrow Q(x)$, который обращается в "истину" при всех тех и только тех $x \in M$, при которых P(x) и Q(x) обращаются оба в истинные или оба в ложные высказывания.

Для его множества истинности имеем:

$$I_{P \leftrightarrow Q} = I_{\bar{P}} \cap I_{\bar{Q}} \cup I_P \cap I_Q$$

Для предикатов справедливы две новые операции, специфические. Они называются - операциями **навешивания кванторов** или **операциями квантификации**. Эти операции соответствуют фразам "для всех" - $\forall$ **квантор общности** и "некоторые" - $\exists$ **квантор существования**.

Операции квантификации превращают одноместный предикат в высказывание.

8.2 Квантор всеобщности

Пусть P(x) – предикат, определенный на множестве M. Под выражением $\forall x P(x)$ понимают высказывание, истинное, когда P(x) истинно для каждого элемента x из множества M, и ложное в противном случае. Это высказывание уже не зависит от x. Соответствующее ему словесное выражение звучит так:

"Для всякого x P(x) истинно ".

Символ $\forall$ называют **квантором всеобщности** (общности). Переменную x в предикате P(x) называют **свободной** (ей можно придавать различные значения из M), в высказывании же $\forall x P(x)$ x называют *связанной квантором всеобщности.*

8.3Квантор существования

Пусть P(x) -предикат определенный на множестве M. Под выражением $\exists x P(x)$ понимают высказывание, которое является истинным, если существует элемент $x \in M$, для которого P(x) истинно, и ложным – в противном случае. Это высказывание уже не зависит от x.

Соответствующее ему словесное выражение звучит так:

"Существует x, при котором P(x) истинно."

Символ $\exists$ называют **квантором существования.** В высказывании $\exists x P(x)$ переменная x связана этим квантором (на нее навешен квантор).

Кванторные операции применяются и к многоместным предикатам. Пусть, например, на множестве M задан двухместный предикат P(x,y). Применение кванторной операции к предикату P(x,y) по переменной x ставит в соответствие двухместному предикату P(x,y) одноместный предикат $\forall x P(x,y)$ (или одноместный предикат $\exists x P(x,y)$), зависящий от переменной y и не зависящий от переменной x. К ним можно применить кванторные операции по переменной y, которые приведут уже к высказываниям следующих видов: $\forall y \forall x P(x,y), \exists y \forall x P(x,y), \forall y \exists x P(x,y), \exists y \exists x P(x,y)$.

Рассмотрим предикат P(x) определенный на множестве $M=\{a_1,\ldots,a_n\}$, содержащем конечное число элементов. Если предикат P(x) является тождественно - истинным, то истинными будут высказывания $P(a_1), P(a_2), \ldots, P(a_n)$. При этом истинными будут высказывания $\forall x P(x)$ и конъюнкция $P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n)$. Если же хотя бы для одного элемента $a_k \in M$ $P(a_k)$ окажется ложным, то ложными будут высказывание $\forall x P(x)$ и конъюнкция $\bigwedge_{i=1}^{n} P(a_i)$. Следовательно, справедлива равносильность

$$\forall x P(x) \equiv P(a_1) \wedge P(a_2) \wedge \ldots \wedge P(a_n) = \bigwedge_{i=1}^{n} P(a_i).$$

Пример: Высказывание *(Экзюпери)*

"Ты любишь потому, что ты любишь. Не существует причин, чтобы любить." можно записать в виде:

$$\mathbf{A} \Rightarrow \mathbf{A}. \quad \neg \exists \mathbf{B},$$

где **A** - *"ты любишь"*, **B** - *"причины любви"*.

✓ **Замечание** Выражение *"существует точно одно X такое, что..."* называется **квантором существования и единственности** и обозначается символом: $\exists!$ X.

8.4 Численные кванторы

В математике часто встречаются выражения вида “по меньшей мере n” (“хотя бы n”), “не более чем n”, “n и только n” (“ровно n”), где n – натуральное число.

Эти выражения, называемые **численными кванторами**, имеют чисто логический смысл; они могут быть заменены равнозначными выражениями, не содержащими числительных и состоящими только из логических терминов и знака $\equiv$ или $\sim$, означающего тождество (совпадение) объектов.

Пусть n=1. Предложение “По меньшей мере один объект обладает свойством P” имеет тот же смысл, что и предложение “Существует объект, обладающий свойством P”, т.е.

$$\exists x(P(x)).(*)$$

Предложение “не более чем один объект обладает свойством P” равнозначно предложению “Если есть объекты, обладающие свойством P, то они совпадают”, т.е.

$$\forall x \forall y(P(x) \wedge P(y) \Rightarrow x \equiv y).(**)$$

Предложение “один и только один объект обладает свойством P” равнозначно конъюнкции вышеуказанных предложений (*) и (**).

8.5 Отрицание предложений с кванторами

Известно, что часто для отрицания некоторого предложения достаточно предпослать сказуемому этого предложения отрицательную частицу “не”. Предложения “Все птицы летают ” и “Все птицы не летают ” не являются отрицаниями друг друга, т. к. они оба ложны. Предложения “ Некоторые птицы летают ” и “ Некоторые птицы не летают ” не являются отрицанием друг друга, т. к. они оба истинны.

Таким образом , предложения , полученные добавлением частицы “не” к сказуемому предложений “Все x суть P” и “Некоторые x суть P” не являются отрицаниями этих предложений.

Универсальным способом построения отрицания данного предложения является добавление словосочетания "наверно, что" в начале предложения.

Таким образом, отрицанием предложения "Все птицы летают" является предложение "Неверно, что все птицы летают"; но это предложение имеет тот же смысл, что и предложение "Некоторые птицы не летают".

Отрицанием предложения "Некоторые птицы летают" является предложение "Неверно, что некоторые птицы летают", которое имеет тот же смысл, что и предложение "Все птицы не летают".

Условимся отрицание предложения $\forall x(P(x))$ записывать как $\overline{\forall}x(P(x))$, а отрицание предложения $\exists x(P(x))$ – как $\overline{\exists}x(P(x))$. Очевидно, что предложение $\overline{\forall}x(P(x))$ имеет тот же смысл, а следовательно, то же значение истинности, что и предложение $\exists x(\overline{P(x)})$, а предложение $\overline{\exists}x(P(x))$ – тот же смысл, что $\forall x(\overline{P(x)})$.

Иначе говоря, $\overline{\forall}x(P(x))$ равносильно $\exists x(\overline{P(x)})$; $\overline{\exists}x(P(x))$ равносильно $\forall x(\overline{P}(x))$.

Кванторы общности и существования называют *двойственными* относительно друг друга. Выясним теперь, как строить отрицание предложения, начинающегося с нескольких кванторов, например, такого: $\forall x\exists y\forall z(P(x,y,z))$.

Последовательно применяя сформулированное выше правило, получим:

$\overline{\forall}x\exists y\forall z(P(x,y,z))$ равносильно $\exists x(\overline{\exists}y\forall z(P(x,y,z))$, что равносильно $\exists x\forall y(\overline{\forall}z(P(x,y,z)))$, что равносильно $\exists x\forall y\exists z(\overline{P}(x,y,z))$.

➢ **Определение** Присоединение квантора с переменной к предикатной формуле называется *навешивание* квантора на переменную x. Переменная при этом называется *связанной* и вместо нее подставлять константы уже нельзя.

Если квантор навешивается на формулу с несколькими переменными, то он уменьшает число несвязанных переменных в этой формуле.

Переменную x в предикате P(x) называют **свободной** (ей можно придавать различные значения из M), в высказывании же $\forall x P(x)$ x называют **связанной** квантором всеобщности.

Переменная, на которую навешивается квантор называется **связанной**.

Выражение, на которое навешивается квантор, называется областью действия квантора.

Пример $\forall x\, P(x)$ **- « из всякого положения есть выход»**

$\exists x\, \overline{P(x)}$ **- « существуют безвыходные положения»**

✓ **Замечание** Квантор общности $\forall$ произошел от английского All и обозначается буквой A, перевернутой вверх ногами.

✓ Квантор существования $\exists$ произошел от английского Exist (существовать) и обозначается буквой E, которую вверх ногами переворачивать бесполезно, поэтому ее повернули кругом.

Интересно посмотреть, как ведут себя кванторы в присутствии операции отрицания.

Возьмем отрицание предиката "ВСЕ любят кашу": "НЕ ВЕРНО, что ВСЕ любят кашу". Это равносильно (по закону Де Моргана!) заявлению: "НЕКОТОРЫЕ НЕ любят кашу.

$$\overline{\forall x\, P(x)} \equiv \exists x \overline{P(x)}$$

То есть отрицание "задвинули" за квантор, в результате чего квантор сменился на противоположный.

Кванторы общности и существования называют **двойственными** относительно друг друга. Из формализованных языков математики язык предикатов – самый близкий к естественному. Поэтому работы по искусственному интеллекту тяготеют к использованию этого языка. Примером тому язык (логического) программирования ПРОЛОГ - ПРОграммирование на ЛОГике.

Вот некоторые "классические примеры"несоответствия языка предикатов и естественного языка.

Пример высказывание "Собакам и кошкам вход воспрещен".

Конструкция "ДЛЯ ВСЕХ иксов справедливо: ЕСЛИ икс - собака И икс - кошка, ТО иксу вход запрещен"

Ясно что таких иксов, которые бы были одновременно собакой и кошкой не существует! Как, впрочем, и таких игреков. Поэтому "ДЛЯ ВСЕХ иксов справедливо: ЕСЛИ икс - собака ИЛИ икс - кошка, ТО иксу вход запрещен"

8.6 Операции навешивания кванторов

Пусть на множестве $\mathbf{M} \neq \varnothing$ задан одноместный предикат p(x).

➢ **Определение** Выражение вида $\forall x$ **p(x)** на множестве **M** представляет собой истинное высказывание, тогда и только тогда, когда **p(x)** истинно для любого элемента $x \in M$

➢ **Определение.** Выражение $\exists x$ **p(x)** на множестве **M** представляет собой истинное высказывание, тогда и только тогда, когда **p(x)** - истинно хотя бы для одного элемента из этого множества. Очевидно,

если $p(x)= 0$, то $\exists x\ p(x) = 0$; и если $p(x)<>0$, то $\exists x\ p(x) = 1$

Пусть на множестве $\mathbf{M} \neq \varnothing$ задан двуместный предикат **p(x,y)**.

➢ **Определение.** Выражение $\forall x$ **p(x,y)** при $y_0 \in M$ представляет собой высказывание $\forall x\ p(x,y_0) = 1$ (истинное высказывание) тогда и только тогда, когда $p(x,y_0)$ - истинно для любого элемента $x \in M$

➢ **Определение.** Выражение $\exists x$ **p(x,y)** при заданном $y_0 \in M$ представляет высказывание $\exists x\ p(x,y_0) = 1$ (истинное высказывание), тогда и только тогда, когда $p(x,y_0)$ истинно хотя бы для одного элемента из множества **M**

Таким образом, операции навешивания кванторов (всеобщности и существования) к двуместным предикатам приводит к одноместному предикату, т.е.:

$\mathbf{p(y) = \forall x\, p(x,y_0)}$ $\mathbf{p(y) = \exists x\, p(x,y_0)}$

Пример Пусть задан двуместный предикат $\mathbf{p(x,y) = \exists x\ (x<y)}$, где $\mathbf{x,y \in R}$

Берем произвольный элемент $\mathbf{y_0}$ из множества **M**, подставляя в данный предикат, получим одноместный предикат: $\mathbf{p(x) = (x<y_0)}$

Тогда выражение $\mathbf{\exists x\ (x<y_0)}$ является истинным высказыванием, так как во множестве действительных чисел всегда для произвольного элемента из этого же множества, найдется элемент меньше его

Если взять предикат вида $\mathbf{p(x,y) = \forall x p(x<y)}$, где $\mathbf{x,y \in R}$, то он является ***ложным*** во множестве **R.**

Пример $P(x,y)$ - « x любит y « - двуместный предикат.

$\forall x \exists y$ - « для любого человека существует y – человек, которого он любит»

$\forall y \exists x$ - « для любого человека существует x – человек, которого он любит»

$\exists y \forall x$ « существует человек, которого любят все»

$\exists x \forall y$ « существует человек, который любит всех»

$\forall x\ \forall y$ все люди любят всех людей»

$\exists y \exists x$ существует человек, который кого-то любит»

8.7 Свойства кванторов

1. $\forall x \forall y\ P(x,y) = \forall y \forall x\ P(x,y)$
   $\exists x \exists y\ P(x,y) = \exists y \exists x\ P(x,y)$ **коммутативность одноименных кванторов.**
2. $\exists x \forall y\ P(x,y) \to \forall y \exists x\ P(x,y)$ обратное не верно
3. $\exists y \forall x\ P(x,y) \to \forall x \exists y\ P(x,y)$

Пример $\forall x \exists y \quad x + y = 0$ истина

$\exists y \forall x \quad x + y = 0$ ложь

Как и в логике высказываний, в логике предикатов имеются эквивалентные соотношения, позволяющие преобразовывать предикатные формулы.

Перестановка кванторов общности и существования меняет смысл.

Теорема (основные законы, содержащие кванторы)

$$\left.\begin{array}{l}.\forall x\forall y P(x,y) \equiv \forall y\forall x P(x,y)\\ .\exists x\exists y P(x,y) \equiv \exists y\exists x P(x,y)\end{array}\right\}$$ - коммутация одноименных кванторов

$$\left.\begin{array}{l}.\forall_\lambda (P(\lambda) \wedge Q(\lambda)) \equiv \forall_\lambda P(\lambda) \wedge \forall_\lambda Q(\lambda)\\ .\exists x(P(x) \vee Q(x)) \equiv \exists x P(x) \vee \exists x Q(x)\end{array}\right\}$$ - дистрибутивные законы для кванторов

$$\left.\begin{array}{l}.\forall x(P(x) \vee Q(y)) \equiv \forall x P(x) \vee \forall x Q(y)\\ .\exists x(P(x) \wedge Q(y)) \equiv \exists x P(x) \wedge \exists x Q(y)\end{array}\right\}$$ - законы ограничения действия кванторов

$$.\exists y\forall x P(x,y) \rightarrow \forall x\exists y P(x,y) \equiv 1$$

Глава 9 Понятие формулы логики предикатов

В логике предикатов будем пользоваться следующей символикой:

1. Символы p, q, r, …- **переменные высказывания,** принимающие два значения: 1- истина , 0 – ложь.

2. **Предметные переменные**– x, y, z, … , которые пробегают значения из некоторого множества M;

x^0, y^0, z^0 – **предметные константы**, т. е. значения предметных переменных.

3. P(·), Q(·), F(·), … - **одноместные предикатные переменные;**

Q(·,·,…,·), R(·,·, …,·) – **n-местные предикатные переменные**.

$P^0(\cdot)$, $Q^0(\cdot,\cdot, \ldots,\cdot)$ – **символы постоянных предикатов**.

4. **Символы логических операций**: $\wedge,\vee,\rightarrow,-$.

5. **Символы кванторных операций:** $\forall x, \exists x$.

6. **Вспомогательные символы**: скобки, запятые.

➢ **Определение Формулы логики предикатов:**

1. Каждое высказывание как переменное, так и постоянное, является формулой (элементарной).

2. Если F(·,·, …,·) – n-местная предикатная переменная или постоянный предикат, а x_1, x_2,…, x_n– предметные переменные или предметные постоянные (не обязательно все различные), то $F(x_1, x_2,\ldots, x_n)$ есть формула. Такая формула называется **элементарной**, в ней предметные переменные являются свободными, не связанными кванторами.

3. Если A и B – формулы, причем, такие, что одна и та же предметная переменная не является в одной из них связанной, а в другой – свободной, то слова $A \vee B, A \wedge B, A \rightarrow B$ есть формулы. В этих формулах те переменные, которые в исходных формулах были свободны, являются свободными, а те, которые были связанными, являются связанными.

4. Если A – формула, то $\overline{A}$ – формула, и характер предметных переменных при переходе от формулы A к формуле $\overline{A}$ не меняется.

5. Если A(x) – формула, в которую предметная переменная x входит свободно, то слова $\forall xA(x)$ и $\exists xA(x)$ являются формулами, причем, предметная переменная входит в них связанно.

6. Всякое слово, отличное от тех, которые названы формулами в пунктах 1 – 5, не является формулой.

Пример P(x) и Q(x,y) – одноместный и двухместный предикаты, а q, r – переменные высказывания, то формулами будут, например, слова (выражения):

$$q, P(x), P(x) \wedge Q(x^0, y), \forall xP(x) \to \exists xQ(x, y), \overline{(Q(x, y) \vee q)} \to r$$

Не является формулой, например, слово:

$$\forall xQ(x, y) \to P(x).$$

Здесь нарушено условие п.3, так как формулу $\forall xQ(x, y)$ переменная x входит связанно, а в формулу P(x) переменная x входит свободно.

Из определения формулы логики предикатов ясно, что всякая формула алгебры высказываний является формулой логики предикатов.

9.1 Равносильные формулы логики предикатов

Определение Две формулы логики предикатов A и B называются **равносильными** на области M, если они принимают одинаковые логические значения при всех значениях входящих в них переменных, отнесенных к области M.

Все равносильности алгебры высказываний будут верны, если в них вместо переменных высказываний подставить формулы логики предикатов. Но, кроме того, имеют место равносильности самой логики предикатов. Рассмотрим основные из этих равносильностей.

Пусть A(x) и B(x) – переменные предикаты, а C – переменное высказывание (или формула, не содержащая x). Тогда имеют место равносильности:

1. $\overline{\forall x A(x)} \equiv \exists x \overline{A(x)}$.

« не верно, что для любого x $A(x)$ истинно» эквивалентно «найдется x, для которого $A(x)$ ложно».

Равносильность означает тот простой факт, что, если не для всех x истинно A(x), то существует x, при котором будет истиной $\overline{A}(x)$.

2. $\overline{\exists x A(x)} \equiv \forall x \overline{A(x)}$.

Равносильность 2 означает тот простой факт, что, если не существует x, при котором истинно A(x), то для всех x будет истиной $\overline{A}(x)$.

Формулы 1-2 называются **правила переноса кванторов через отрицание или законы де Моргана для кванторов**.

3. $\forall x A(x) \equiv \overline{\exists x \overline{A(x)}}$.

4. $\exists x A(x) \equiv \overline{\forall x \overline{A(x)}}$.

Равносильности 3 и 4 получаются из равносильностей 1 и 2, соответственно, если от обеих их частей взять отрицания и воспользоваться законом двойного отрицания.

5. $\forall x A(x) \wedge \forall x B(x) \equiv \forall x [A(x) \wedge B(x)]$

«квантор можно вносить и выносить за скобки в конъюнкции»

6. $C \wedge \forall x B(x) \equiv \forall x [C \wedge B(x)]$.

7. $C \vee \forall x B(x) \equiv \forall x [C \vee B(x)]$

8. $C \to \forall x B(x) \equiv \forall x [C \to B(x)]$

9. $\forall x [B(x) \to C] \equiv \exists x B(x) \to C$.

Для формул 6-9 : постоянное высказывание можно вносить под знак квантора всеобщности и выносить из под знака в конъюнкции, дизъюнкции и импликации.

10. $\exists x [A(x) \vee B(x)] \equiv \exists x A(x) \vee \exists x B(x)$.

квантор существования можно вносить и выносить за скобки в дизъюнкции»

11. $\exists x [C \vee B(x)] \equiv C \vee \exists x B(x)$.

12. $\exists x[C \wedge B(x)] \equiv C \wedge \exists x B(x)$.
13. $\exists x A(x) \wedge \exists y B(y) \equiv \exists x \exists y[A(x) \wedge B(y)]$.
14. $\exists x[C \to B(x)] \equiv C \to \exists x B(x)$.
15. $\exists x[B(x) \to C] \equiv \forall x B(x) \to C$.

Законы логических операций

1. $\overline{\forall x P(x)} \equiv \exists x \overline{P}(x)$.
2. $\overline{\exists x P(x)} \equiv \forall x \overline{P}(x)$.
3. $\forall x P(x) \equiv \overline{\exists x \overline{P}(x)}$.
4. $\exists x P(x) \equiv \overline{\forall x \overline{P}(x)}$.
5. $\forall x[P(x) \wedge Q(x)] \equiv \forall x P(x) \wedge \forall x Q(x)$.
6. $\exists x[P(x) \vee Q(x)] \equiv \exists x P(x) \vee \exists x Q(x)$.
7. $\forall x \forall y P(x, y) \equiv \forall y \forall x P(x, y)$.
8. $\exists x \exists y P(x, y) \equiv \exists y \exists x P(x, y)$.
9. $\exists x \forall y P(x, y) \Rightarrow \forall y \exists x P(x, y)$.
10. $\forall x P(x) \vee \forall x Q(x) \Rightarrow \forall x[P(x) \vee Q(x)]$.
11. $P(y) \Rightarrow \exists x P(x) \equiv \exists x[P(y) \Rightarrow P(x)]$.
12. $C \Rightarrow \exists x P(x) \equiv \exists x[C \Rightarrow P(x)]$.
13. $C \wedge \forall x P(x) \equiv \forall x[C \wedge P(x)]$.
14. $C \vee \forall x P(x) \equiv \forall x[C \vee P(x)]$.
15. $C \Rightarrow \forall x P(x) \equiv \forall x[C \Rightarrow P(x)]$.
16. $\exists x[C \vee P(x)] \equiv C \vee \exists x P(x)$.
17. $\exists x[C \wedge P(x)] \equiv C \wedge \exists x P(x)$.
18. $\exists x P(x) \wedge \exists y Q(y) \equiv \exists x \exists y[P(x) \wedge Q(y)]$.
19. $\exists x[P(x) \Rightarrow C] \equiv \forall x P(x) \Rightarrow C$.
20. $\forall x P(x) \vee \forall x Q(x) \equiv \forall x P(x) \vee \forall y Q(y) \equiv$
$\equiv \forall x[P(x) \vee \forall y Q(y)] \equiv \forall x \forall y[P(x) \vee Q(y)]$.

21.

$$\exists x P(x) \wedge \exists x Q(x) \equiv \exists x P(x) \wedge \exists y Q$$

$$\equiv \exists x[P(x) \wedge \exists y Q(y)] \equiv \exists x \exists y[P(x$$

22. $\exists x[P(x) \wedge Q(x)] \Rightarrow \exists x P(x) \wedge \exists x Q(x)$.

23. $\forall x[P(x) \Rightarrow Q(x)] \Rightarrow [\forall x P(x) \Rightarrow \forall x Q(x)$.

24. $\forall x P(x) \Rightarrow \exists x P(x)$.

25. $\forall x[P(x) \Rightarrow P(y)] \equiv \exists x P(x) \Rightarrow P(y)$.

26. $\forall x[P(x) \Rightarrow C] \equiv \exists x P(x) \Rightarrow C$.

Пример Доказать тождественную истинность формул:

а) $\forall x(F(x) \to F(x))$;

б) $\forall x \forall y \forall z((F(x) \to G(y)) \wedge (G(y) \to H(z)) \to (F(x) \to H(z)))$.

Решение

Будем преобразовывать бескванторную часть формул, используя основные логические законы.

а) $F(x) \to F(x) \equiv \overline{F(x)} \vee F(x) \equiv 1$.

Значит, исходная формула тождественно истинна.

б) $(F(x) \to G(y)) \wedge (G(y) \to H(z)) \to (F(x) \to H(z)) \equiv$

$\equiv (\overline{F(x)} \vee G(y)) \wedge (\overline{G(y)} \vee H(z)) \to (\overline{F(x)} \vee H(z)) \equiv$

$\equiv \overline{(\overline{F(x)} \vee G(y)) \wedge (\overline{G(y)} \vee H(z))} \vee (\overline{F(x)} \vee H(z)) \equiv$

$\equiv \overline{\overline{F(x)} \vee G(y)} \vee \overline{\overline{G(y)} \vee H(z)} \vee (\overline{F(x)} \vee H(z)) \equiv$

$\equiv (F(x) \wedge \overline{G(y)}) \vee (G(y) \wedge \overline{H(z)}) \vee \overline{F(x)} \vee H(z) \equiv$

$\equiv (((F(x) \wedge \overline{G(y)}) \vee \overline{F(x)}) \vee ((G(y) \wedge \overline{H(z)}) \vee H(z)) \equiv$

$\equiv ((F(x) \vee \overline{F(x)}) \wedge (\overline{G(y)} \vee \overline{F(x)})) \vee$

$\vee ((G(y) \vee H(z)) \wedge (\overline{H(z)} \vee H(z))) \equiv$

$\equiv (1 \wedge (\overline{G(y)} \vee \overline{F(x)})) \vee ((G(y) \vee H(z)) \wedge 1) \equiv$

$\equiv (\overline{G(y)} \vee \overline{F(x)}) \vee (G(y) \vee H(z)) \equiv (\overline{G(y)} \vee G(y)) \vee \overline{F(x)} \vee H(z) \equiv$

$\equiv 1 \vee \overline{F(x)} \vee H(z) \equiv 1$

9.2 Значение формулы логики предикатов

О логическом значении формулы логики предикатов можно говорить лишь тогда, когда задано множество M, на котором определены входящие в эту формулу предикаты.

Логическое значение формулы логики предикатов зависит от значений трех видов переменных:

1) значений входящих в формулу переменных высказываний,

2) значений свободных предметных переменных из множества M,

3) значений предикатных переменных.

При конкретных значениях каждого из трех видов переменных формула логики предикатов становится высказыванием, имеющим истинное или ложное значение.

Рассмотрим формулу

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

в которой двухместный предикат P(x, y) определен на множестве MxM, где M={0,1,2,…,n,…}, т.е. MxM=NxN.

В формулу входит переменный предикат P(x,y), предметные переменные x,y,z, две из которых y и z – связанные кванторами, а x – свободная.

Возьмем за конкретное значение предиката P(x,y) фиксированный предикат P^0(x,y): “x<y”, а свободной переменной x придадим значение $x^0 = 5 \in M$.

Тогда при значениях y, меньших x^0=5, предикат $P^0(x^0,y)$ принимает значение “ложь”, а импликация $P(x, y) \to P(y, z)$ при всех $z \in M$ принимает значение “истина”, т.е. высказывание $\exists y \forall z (P^0(x, y) \to P^0(y, z))$ имеет значение “истина”.

Контрольные вопросы

1. Что такое формула логики предикатов?
2. Что такое подформула логики предикатов?

3. Дайте определение свободной и связанной переменной формулы логики предикатов?
4. Привести примеры формул. Указать все свободные и связанные переменные этих формул.
5. Дать определение истинности формулы логики предикатов Привести примеры.
6. Что такое логическое следствие в логике предикатов. Дать определение тождественно ложной формулы логики предикатов.
7. Дать определение доказуемой и выводимой из множества формул формулы исчисления предикатов, тавтологии исчисления предикатов. Привести примеры тавтологий исчисления предикатов.

Задачи для самостоятельного решения

1. Записать фразы в виде формул логики предикатов, указав области определения используемых предикатов:

а) «Если произведение двух чисел делится на простое число, то на него делится хотя бы один из сомножителей»;

б) «Через всякую точку, не лежащую на прямой, можно провести прямую, перпендикулярную данной»;

в) «Через каждые две точки можно провести прямую, причем, если эти точки различны, то такая прямая единственна»;

г) «Когда у некоего деятеля денег в избытке, он может воспевать что угодно, в том числе и отсутствие денег»;

д) «Всякий кулик свое болото хвалит, а чужое хает».

2. Пусть f^1 – одноместный, g^2 – двухместный, h^3 – трехместный функциональные символы. Являются ли термами следующие слова:

а) $f^1(g^2(v_0,v_1))$;

б) $g^2(f^1(v_2),h^3(v_0,v_1,v_2))$;

в) $f^1(g^2(v_0),h^3(v_0,v_1,v_2))$?

3. Пусть f^1, g^2, h^3 – те же, что и в задаче 2., P^1 – одноместный, Q^3 – трехместный предикатные символы. Являются ли формулами следующие слова:

а) $Q^3(v_0, f^1(v_1), h^3(v_0, v_1, v_2))$;

б) $(P^1(v_0) \to \forall v_1(Q^3(v_0, v_1, v_2) \wedge P^1(g^2(v_0, v_1))))$;

в) $Q^3(P_1(v_0), f^1(v_1), f^1(v_2))$;

г) $f^1(h^3(v_0, v_1, v_2))$?

4. Выписать все подформулы формулы:

а) $Q^2(f^1(v_0), g^2(v_0, v_1))$;

б) $(\exists v_0 Q^2(v_0, v_1) \to \overline{P^1(g^2(v_0, v_1)) \wedge \forall v_2 P^1(v_2)})$.

5. а) Найти такие два замещения предикатного символа F конкретным предикатом в формуле $\exists x F(x) \to \forall x F(x)$, чтобы при одном замещении получилось истинное предложение, при другом – ложное.

б) То же для формулы $\exists x(F(x) \vee G(x)) \to \forall x H(x)$.

в) То же для формулы $\forall x \exists y F(x, y) \to \exists y \forall x F(x, y)$.

6. Доказать тождественную ложность формул:

а) $\exists x((F(x) \to \overline{F(x)}) \wedge (\overline{F(x)} \to F(x)))$;

б) $\exists x \exists y((F(x) \to F(y)) \wedge (F(x) \to \overline{F(y)}) \wedge F(x))$.

7. Доказать, что:

а) $\forall x(F(x) \wedge G(x)) \equiv \forall x F(x) \wedge \forall x G(x)$;

б) $\exists x(F(x) \vee G(x)) \equiv \exists x F(x) \wedge \exists x G(x)$.

Глава 10 Нормальные формы ЛП

В логике предикатов, как и в логике высказываний, формулы могут иметь нормальную форму. При этом, используя равносильности логики предикатов, каждую формулу логики предикатов можно привести к нормальной форме. В логике предикатов различают два вида нормальных форм: приведенную и предваренную.

➢ **Определение** Формула логики предикатов имеет **приведенную нормальную форму**, если она содержит только операции конъюнкции, дизъюнкции и кванторные операции, а операция отрицания отнесена к элементарным формулам.

Пример

$$(\exists x P(x) \to \forall y Q(y)) \to R(z) \overset{18}{\equiv}$$

$$\equiv \overline{\exists x P(x)} \vee \forall y Q(y) \vee R(z) \overset{17}{\equiv}$$

$$\equiv \overline{\overline{\exists x P(x)}} \wedge \overline{\forall y Q(y)} \vee R(z) \overset{1,31}{\equiv}$$

$$\overset{1,31}{\equiv} \exists x P(x) \wedge \exists y \overline{Q(y)} \vee R(z)$$

Получили приведенную нормальную форму исходной формулы.

Среди нормальных форм формул логики предикатов выделяют так называемую **предваренную** (префиксную) нормальную форму (ПНФ). В ней кванторные операции либо полностью отсутствуют, либо они используются после всех операций алгебры логики, т.е.

ПНФ имеет вид

$$(\sigma_{x1})(\sigma_{x2})\ldots(\sigma_{xn})A(x_1, x_2, \ldots, x_m), n \le m,$$

где под символом σ_{xi} понимается один из кванторов $\forall x_i$ или $\exists x_i$, а формула А кванторов не содержит.

✓**Замечание** Бескванторная формула также считается предваренной.

10.1 Алгоритм получения (приведения) ПНФ

Пусть задана формула A логики предикатов. Формула B называется **предваренной нормальной формой** формулы A, если она удовлетворяет ниже перечисленным требованиям:

1. Формулы A и B равносильны.

2. Формула B удовлетворяет следующим условиям:

а) используются логические операции ┐, v, &, при этом отрицание применяется только в атомарных формулах;

б) операции кванторов следуют за операциями алгебры ¬, &, v.

Алгоритм получения предваренной нормальной формы представлен ниже.

Шаг 1. Исключить связки эквивалентности (~) и импликации (→).

Формула $x \sim y$ заменяется на $(x \to y)\ \&\ (x \to y)$, а формула $A \to B$ заменяется на $(\bar{A}\ \mathrm{v}\ B)$.

Шаг 2. Переименовать, если необходимо, связанные переменные таким образом, чтобы никакая переменная не имела бы одновременно свободных и связанных вхождений. Это условие рассматривается и по отношению к подформулам.

Шаг 3. Удалить те квантификации, область действия которых не содержит вхождений квантифицированной переменной.

Шаг 4. Перенести отрицания внутри формулы в соответствия со следующими правилами:

$$\frac{A \to B,\ A}{B};\quad \overline{\forall x A} \sim \exists x \bar{A};\quad \overline{A \& B} \sim \bar{A} \& \bar{B};\quad \bar{\bar{A}} \sim A.$$

Шаг 5. Перенести все квантификации в начало формулы

Пример. Найти ПНФ формулы $x \to (y \to xy)$.

Решение.

Шаг 1: $\forall x(P(x) \& \forall y \exists x(\bar{\bar{Q}}(x,y) \mathrm{v} \forall z R(a,x,y)))$

Шаг 2: $\forall x(P(x) \& \forall y \exists u(\bar{\bar{Q}}(u,y) \mathrm{v} \forall z R(a,u,y)))$

Шаг 3: $\forall x(P(x) \& \forall y \exists x(\overline{\overline{Q}}(x,y) \vee R(a,x,y)))$

Шаг 4: $\forall x(P(x) \& \forall y \exists x(Q(x,y) \vee R(a,x,y)))$

Шаг 5: $\forall x \forall y(p(x) \& \exists u(Q(u,y) \vee R(a,u,y)))$.

$\forall x \forall y \exists u(P(x) \& (Q(u,y) \vee R(a,u,y)))$.

Предваренная нормальная форма включает в себя префикс, образованный кванторами $\forall$ и $\exists$ и матрицу, под которой понимается формула, не содержащая квантификаций.

Пример Привести формулы к ПНФ:

а) $\overline{\exists x F(x,z)} \wedge (G(x,y) \to \forall x H(z,x))$;

б) $\neg(\exists x F(x,y,z) \to \forall x F(x,x,x)) \vee F(x,z,y)$.

Решение

Преобразуем формулы, устраняя «лишние» логические связки и «опуская» отрицания на предикаты, а затем последовательно «вытаскиваем» кванторы из-под логических связок с помощью преобразований типа А или типа В, при необходимости применяя переименование переменных.

а) $\overline{\exists x F(x,z)} \wedge (G(x,y) \to \forall x H(z,x)) \equiv$ (B) $\equiv$

$\equiv \forall x \overline{F(x,z)} \wedge (\overline{G(x,y)} \vee \forall x H(z,x)) \equiv$ (A) $\equiv$

$\equiv \forall x \overline{F(x,z)} \wedge \forall t(\overline{G(x,y)} \vee H(z,t)) \equiv$ (A) $\equiv$

$\equiv \forall t(\forall x \overline{F(x,z)} \wedge (\overline{G(x,y)} \vee H(z,t))) \equiv$ (A) $\equiv$

$\equiv \forall u \forall t(\overline{F(u,z)} \wedge (\overline{G(x,y)} \vee H(z,t)))$.

б) $\neg(\exists x F(x,y,z) \to \forall x F(x,x,x)) \vee F(x,z,y) \equiv$

$\equiv \neg(\overline{\exists x F(x,y,z)} \vee \forall x F(x,x,x)) \vee F(x,z,y) \equiv$

$\equiv (\exists x F(x,y,z) \wedge \overline{\forall x F(x,x,x)}) \vee F(x,z,y) \equiv$ (A,e) $\equiv$

$\equiv (\exists Ц(F(Ц,y,z) \wedge \exists x \overline{F(x,x,x)})) \vee F(x,z,y) \equiv$ (e) $\equiv$

$\equiv \exists Ц(\exists x(F(Ц,y,z) \wedge F(x,x,x))) \vee F(x,z,y)) \equiv$ (e) $\equiv$

$\equiv \exists Ц \exists t(F(Ц,y,z) \wedge F(t,t,t) \vee F(x,z,y))$.

Задачи для самостоятельного решения

1. Привести к ПНФ:

а) $\overline{\exists x F(x,z)} \wedge (G(x,y) \to \forall z H(z,x))$;

б) $\forall x \overline{\exists y F(x,y,z)} \wedge \exists x \exists y G(y,x) \to \exists z H(z)$;

в) $\neg(\exists x F(x,y,z) \to \forall x F(x,x,x)) \vee F(x,y,z))$;

г) $\neg(\exists z \forall x (G(x,y) \wedge \overline{F(x,z)} \to \forall y H(y,z))$;

д) $\forall x (\overline{F(x)} \to \forall y (\overline{F(y)} \to \forall z (\overline{F(z)} \to \forall u \overline{F(u)})))$;

е) $F(x_1,x_2) \wedge \forall x_1 \forall x_2 G(x_1,x_2,x_3) \wedge \overline{\exists x_1 F(x_1,x_2)}$;

ж) $\forall x_1 \forall x_2 ... \forall x_n F(x_1,...,x_n) \to \exists x_1 ... \exists x_n F(x_1,...,x_n)$;

з) $\forall x_1 F(x_1,...,x_n) \to (\forall x_2 F(x_1,...,x_n) \to ...$

$...(\forall x_n F(x_1,...,x_n) \to G(y))...)$.

10.2 Скулемовские функции

Приведение формулы ЛП к сколемовской форме (сколемизация) призвано обеспечить дальнейшее упрощение логических представлений и облегчить введение процедур машинной обработки в ЛП.

Отправной точкой сколемизации является предваренная нормальная форма, матрица которой приведена к конъюнктивной нормальной форме (КНФ).

Цель сколемизации - исключение ∃- квантификаций. Сколемовская форма получается путем применения следующей процедуры:

1) сопоставить каждой ∃- квантифицированной переменной список ∀- квантифицированных переменных, предшествующих ей, а также некоторую еще не использованную функциональную константу, число мест, у которой равно мощности списка. Данная константа будет представлять сколемовскую функцию;

2) в матрице формулы заменить каждое вхождение каждой ∃-

квантифицированной переменной на некоторый терм. Этот терм является функциональной константой, соответствующей данной переменной и снабженной списком аргументов, также соответствующим той же самой переменной;

3) устранить из формулы все $\exists$ - квантафикации.

Пример Найти сколемовскую форму формулы $\forall x \exists y \forall z \exists \upsilon P(x, y, z, u, \upsilon)$.

Решение

$\forall x \forall z \forall u P(x, f(x), z, u, g(x, z, u))$, где f и g - сколемовские функции.

➢ **Определение Клаузальной** формой называется такая сколемовская форма, матрица которой приведена к КНФ. Любая сколемовская форма допускает эквивалентную клаузальную форму.

Пример .

$\exists x \forall y P(x,y) \vee \overline{\forall x \exists y Q(x,y)} \equiv$

$\equiv \exists x \forall y P(x,y) \vee \exists x \forall y \overline{Q(x,y)} \equiv^{36}$

$\equiv \exists x [\forall y P(x,y) \vee \forall y \overline{Q(x,y)}]$

обозначим в предикате Q переменную y через z

$\equiv \exists x [\forall y P(x,y) \vee \forall z \overline{Q(x,z)}] \equiv^{53} \exists x \forall y \forall z (P(x,y) \vee \overline{Q(x,z)})$

Пример

$\overline{(\exists x \forall y P(x,y) \wedge \exists x \forall y Q(x,y))} \equiv^{16}$

$\equiv \overline{\exists x \forall y P(x,y)} \vee \overline{\exists x \forall y Q(x,y)} \equiv^{31,32}$

$\equiv \forall x \exists y \overline{P(x,y)} \vee \forall x \exists y \overline{Q(x,y)}$

обозначим в предикате Q переменную x через z

$\equiv \forall x \exists y \overline{P(x,y)} \vee \forall z \exists y \overline{Q(z,y)}] \equiv^{53}$

$\equiv \forall x \forall z (\exists y \overline{P(x,y)} \vee \exists y \overline{Q(z,y)}) \equiv^{36}$ – ПНФ.

$\equiv \forall x \forall z \forall y (\overline{P(x,y)} \vee \overline{Q(z,y)})$

Пример

$$\forall x\forall y(\exists z(P(x,z)\wedge Q(y,z))\to\exists uR(x,y,u))\overset{18}{\equiv}$$

$$\equiv\forall x\forall y\overline{(\exists z(P(x,z)\wedge Q(y,z))}\vee\exists uR(x,y,u)\overset{32}{\equiv}$$

$$\equiv\forall x\forall y(\forall z\overline{(P(x,z)\wedge Q(y,z)}\vee\exists uR(x,y,u)\overset{16}{\equiv}$$

$$\equiv\forall x\forall y(\forall z(\overline{P(x,z)}\vee\overline{Q(y,z)})\vee\exists uR(x,y,u)$$

последний предикат не зависит от переменной z

$$\overset{47}{\equiv}\forall x\forall y\forall z(\overline{P(x,z)}\vee\overline{Q(y,z)}\vee\exists uR(x,y,u))\equiv\Big|$$ два первых предиката не зависят от переменной u

$$\Big|\overset{49}{\equiv}\forall x\forall y\forall z\exists u(\overline{P}(x,z)\vee\overline{Q}(y,z)\vee R(x,y,u))$$ - ПНФ.

Пример

$$(\overline{\exists uP(u)}\to\overline{\forall y\forall u\overline{Q(y,u)}})\to\forall xR(x)\overset{18}{\equiv}$$

$$\overline{\overline{(\exists uP(u)}}\vee\overline{\forall y\forall u\overline{Q(y,u)}})\to\forall xR(x)\overset{1}{\equiv}$$

$$\overset{1}{\equiv}\exists uP(u)\vee\overline{\forall y\forall u\overline{Q(y,u)}}\to\forall xR(x)\overset{18}{\equiv}$$

$$\overline{\exists uP(u)\vee\overline{\forall y\forall u\overline{Q(y,u)}}}\vee\forall xR(x)\overset{16}{\equiv}$$

$$\overset{16}{\equiv}\overline{\exists uP(u)}\wedge\overline{\overline{\forall y\forall u\overline{Q(y,u)}}}\vee\forall xR(x)\overset{1}{\equiv}$$

$$\overline{\exists u\overline{P(u)}}\wedge\forall y\forall uQ(y,u)\vee\forall xR(x)\overset{32}{\equiv}$$

$$\overset{32}{\equiv}\forall u\overline{P(u)}\wedge\forall y\forall uQ(y,u)\vee\forall xR(x)\overset{35}{\equiv}$$

$$\forall u(\overline{P(u)}\wedge\forall yQ(y,u))\vee\forall xR(x)\overset{46}{\equiv}$$

$$\overset{46}{\equiv}\forall u\forall y(\overline{P(u)}\wedge Q(y,u))\vee\forall xR(x)\overset{47}{\equiv}$$

$$\forall u\forall y\forall x(\overline{P(u)}\wedge Q(y,u)\vee R(x))\overset{37}{\equiv}$$

$$\overset{37}{\equiv}\forall x\forall y\forall u(\overline{P(u)}\wedge Q(y,u)\vee R(x))-\text{ПНФ}.$$

10.3 Общезначимость и выполнимость формул логики предикатов

➢ **Определение** Формула А логики предикатов называется **выполнимой** в области М, если существуют значения переменных входящих в эту формулу и отнесенных к области М (иначе – существует модель), при которых формула А принимает истинные значения.

➢ **Определение** Формула А логики предикатов называется **общезначимой,** если она тождественна истинна на всякой области (на любой модели).

Все логические законы, представленный в логике высказываний формулами являются общезначимыми формулами логики предикатов.

Общезначимость формулы логики предикатов, например, F обозначается $\vdash$**F**.

Все общезначимые формулы могут быть источниками новых $\vdash$ формул. Например, подставляя в (14) – закон исключенного третьего $x \vee \bar{x}$ – вместо x предикат $P(x_1,\ldots,x_n)$, получаем общезначимую формулу $P(x_1,\ldots,x_n) \vee \overline{P}(x_1,\ldots,x_n)$. При n=1 имеем общезначимую формулу $P(x) \vee \overline{P}(x)$, и, таким образом , $\forall x[P(x) \vee \overline{P}(x)]$- общезначимая формула логики предикатов.

Из тождественно истинной формулы логики высказываний (2) $x \vee y \equiv y \vee x$ подстановкой вместо x предиката P(x, y), а вместо y- предиката Q(x,y) получаем общезначимую формулу $P(x,y) \vee Q(x,y) \equiv Q(x,y) \vee P(x,y)$ и т. д.

➢ **Определение** Формула А логики предикатов называется **тождественно ложной** в области М, если она принимает ложные значения для всех значений переменных, входящих в эту формулу и отнесенных к этой области (иными словами, на данной модели).

Из приведенных определений с очевидностью следует:

1. Если формула А общезначима, то она и выполнима на всякой области (модели).

2. Если формула А тождественно истинна в области М, то она и выполнима в этой области.

3. Если формула А тождественно ложна в области М, то она не выполнима в этой бласти.

4. Если формула А не выполнима, то она тождественно ложна на всякой области (на всякой модели).

5. Для того, чтобы формула А логики предикатов была общезначима, необходимо и достаточно, чтобы ее отрицание было не выполнимо.

6. Для того, чтобы формула А логики предикатов была выполнимой, необходимо и достаточно, чтобы формула $\overline{A}$ была не общезначима.

Пример Определить тип формулы $\exists x \exists y[P(x) \wedge \overline{P}(y)] = F..$

Решение

Пусть P(x): " Число x - четно –" предикат, определенный в $M=N^2$.

Таким образом, рассматриваемая формула на данной модели представляет собой следующее утверждение:" Среди натуральных чисел существуют как четные, так и нечетные ". Очевидно, что это высказывание истинно и, таким образом, на данной модели формула F тождественно истинна.

Однако, если этот же предикат задать на множестве M=NxN,где N – множество четных чисел, то формула F на такой модели окажется тождественно ложной.

Учитывая изложенное, заключаем, что рассматриваемая формула F выполнима (но не общезначима).

Пример Для формулы $\exists y P(x, y, z)$ подобрать модель, на которой она является тождественно истинной (и, таким образом, в целом выполнимой).

Решение

Пусть P(x, x, y): "x·x=y", или иначе "x^2=y" – предикат, определенный на множестве натуральных чисел, т.е. M=N.

Тогда рассматриваемая формула выражает утверждение о существовании натурального квадрата натурального числа, что,

очевидно, является истиной, т.е. на данной модели формула тождественно истинна, что и требовалось доказать.

Пример Определить тип формулы $\forall x P(x, y, x)$.

Решение

Это выполнимая формула. Действительно, если P(x, y, x): "x+y=x" – предикат суммы, то на M=N существует подстановка вместо y соответствующего значения, дающего значение истинности данной формуле. Очевидно, это y=0, поскольку в этом случае получаем "x=x".

Если же P(x, y, x): "xy=x" – предикат произведения, то таким значением y является y=1, так как при нем получаем истинное высказывание $\forall x(x \cdot 1 = x)$.

Но это единственные подстановки, приводящие к верным утверждениям, что и говорит именно о выполнимости данной формулы (но не об ее общезначимости).

Пример Является ли общезначимой формула: $\exists x P(x, y) \rightarrow \forall x P(x, y)$?

Решение

Пусть P(x, y) – предикат порядка (бинарного отношения) "$x \le y$", определенный на конечном множестве натуральных чисел M_1.

Тогда при подстановке в формулу вместо свободной переменной y величины $y = a_{\max} \in M_1$ получим истинное утверждение, а при подстановке любой другой константы из множества M_1 – ложное. Таким образом, рассматриваемая формула не является общезначимой.

Пример Доказать, что формула $\overline{A(y)} \wedge \forall z A(z)$ невыполнима.

Решение

Допустим противное, т.е. что она выполнима. Это означает, что существует такое множество M и такой конкретный предикат A^0 в нем, что когда $y, z \in M$, то данная формула превращается в такой конкретный предикат $B(y) = \overline{A^0(y)} \wedge \forall z A^0(z)$, который, в свою очередь,

превращается в истинное высказывание при всякой подстановке вместо у элементов из множества М.

Возьмем любое $y^0 \in M$. Тогда высказывание $B(y^0) = \overline{A^0(y^0)} \wedge \forall z A^0(z)$ истинно, как мы только что установили. Следовательно, истинны высказывания $\overline{A^0(y^0)}$ и $\forall z A^0(z)$.

Из истинности второго высказывания заключаем, что высказывание $A^0(y^0)$ истинно (поскольку "для всех предметных переменных", как бы они ни обозначались). Но это противоречит истинности первого высказывания $\overline{A^0(y^0)}$.

Таким образом, предположение о выполнимости формулы неверно.

Проблема разрешимости в логике предикатов ставится так же, как и в алгебре логики: существуют ли алгоритмы, позволяющие для любой формулы А логики предикатов установить, к какому типу (классу) она относится, т.е. является ли она общезначимой, выполнимой или тождественно ложной (невыполнимой).

Если бы такой алгоритм существовал, то, как и в алгебре высказываний, он сводился бы к критерию тождественной истинности любой формулы логики предикатов.

Отметим, что, в отличие от алгебры логики, в логике предикатов не применим метод перебора всех вариантов значений переменных, входящих в формулу, так как таких вариантов может быть бесконечное множество.

Теорема (*Теорема Черча*). Не существует алгоритма, который для любой формулы логики предикатов устанавливает, общезначима она или нет.

Пример формула $\forall x[P(x) \wedge \overline{P}(x)]$ является тождественно ложной (невыполнимой) формулой логики предикатов.

Пример Доказать равносильность (логическое тождество):

$$(\exists x \forall y \overline{P}(x,y) \vee \exists u \forall v P(v,u)) \wedge (\forall v \exists u P(v,u) \wedge \exists y \forall x P(x,y)) \equiv$$

$$\equiv \exists x \forall y P(y,x) \wedge \forall u \exists v P(u,v)$$

Заметив, что в каждой из кванторных подформул обе предметные переменные связаны и что, таким образом, они являются высказываниями, введем обозначения:

$A = \forall x \exists y P(x,y) = \forall v \exists u P(v,u) = \forall u \exists v P(u,v)$,

$B = \exists u \forall v P(v,u) = \exists y \forall x P(x,y) = \exists x \forall y P(y,x)$,

или обозначив первую и вторую предметные переменные через n1 и n2, соответственно:

$A = \forall n_1 \exists n_2 P(n_1, n_2)$, $B = \exists n_2 \forall n_1 P(n_2, n_1)$.

В этих обозначениях заданное для рассмотрения тождество будет выглядеть так: $(\overline{A} \vee B) \wedge (A \wedge B) \equiv B \wedge A$.

Произведя равносильные преобразования, можем убедиться в справедливости этого тождества: $(\overline{A} \vee B)AB \overset{6}{\equiv} \overline{A}AB \vee BAB \overset{15,9}{\equiv} ЛВ \vee BA \overset{10}{\equiv} BA$ видим, что это общезначимая формула.

10.4 Применение языка логики предикатов для записи математических предложений

Язык логики предикатов удобен для записи математических предложений и определений. Он дает возможность выражать логические связи между понятиями, записывать определения, теоремы, доказательства. Приведем несколько примеров таких записей.

Пример Определение предела "b" функции $f(x)$, определенной в области E, в точке x_0:

$$b = \lim_{x \to x_0} f(x) \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 \forall x \in E$$

$$0 < |x - x_0| < \delta \to |f(x) - b| < \varepsilon$$

Используя трехместный предикат $P(\varepsilon, \delta, x)$, запишем:

$$b = \lim_{x \to x_0} f(x) \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 \forall x \in E(P(\varepsilon, \delta, x)),$$

где $P(\varepsilon,\delta,x)=(0<|x-x_0|<\delta\to|f(x)-b|<\varepsilon)$.

Пример Определение непрерывности функции в точке.

Функция $f(x)$, определенная на множестве E, непрерывна в точке $x_0\in E$, если $\forall\varepsilon>0\exists\delta>0\forall x\in E(P(\varepsilon,\delta,x))$, где $P(\varepsilon,\delta,x)=(0<|x-x_0|<\delta\to|f(x)-f(x_0)|<\varepsilon)$.

10.5 Прямая, обратная и противоположная теоремы

Рассмотрим четыре теоремы:

$\forall x\in E(P(x)\to Q(x))$,

$\forall x\in E(Q(x)\to P(x))$,

$\forall x\in E(\overline{P}(x)\to\overline{Q}(x))$

$\forall x\in E(\overline{Q}(x)\to\overline{P}(x))$

➢ **Определение** Пара теорем, у которых условие одной является заключением второй, а условие второй является заключением первой, называются ***взаимно обратными друг другу.***

Так, теоремы $\forall x\in E(P(x)\to Q(x))$ и $\forall x\in E(Q(x)\to P(x))$, а также $\forall x\in E(\overline{P}(x)\to\overline{Q}(x))$ и $\forall x\in E(\overline{Q}(x)\to\overline{P}(x))$ - взаимно обратные теоремы. При этом, если одну из них называют *прямой теоремой*, то вторая называется *обратной*.

➢ **Определение** Пара теорем, у которых условие и заключение одной являются отрицанием соответственно условия и заключения другой, называются **взаимно противоположными.**

Так, теоремы $\forall x\in E(P(x)\to Q(x))$ и $\forall x\in E(\overline{P}(x)\to\overline{Q}(x))$, а также $\forall x\in E(Q(x)\to P(x))$ и $\forall x\in E(\overline{Q}(x)\to\overline{P}(x))$ являются взаимно противоположными теоремами.

Пример

Теорема "Если в четырехугольнике диагонали равны, то четырехугольник является прямоугольником " (1)

обратной является теорема

"Если четырехугольник является прямоугольником, то его диагонали равны" (2).

Для теоремы (1) **противоположной** является теорема

Если в четырехугольнике диагонали не равны, то четырехугольник не является прямоугольником " (3), а

для теоремы (2) противоположной является теорема

"Если четырехугольник не является прямоугольником, то его диагонали не равны " (4).

В рассмотренном примере теоремы (1) и (4) являются одновременно ложными, а теоремы (2) и (3) одновременно истинными.

Контрпримером к теореме (1) является равнобочная трапеция.

Вывод: **прямая и обратная теоремы,** вообще говоря**, не равносильны**, т. е. одна из них может быть истинной, а другая – ложной. Однако легко показать, что теоремы (1) и (4), а также (2) и (3) всегда равносильны.

Действительно:
$$\forall x \in E(P(x) \to Q(x)) \overset{18}{\equiv} \forall x \in E(\overline{P}(x) \vee Q(x)) \overset{1,2}{\equiv}$$
$$\equiv \forall x \in E(\overline{\overline{Q}}(x) \vee \overline{P}(x)) \overset{18}{\equiv} \forall x \in E(\overline{Q}(x) \to \overline{P}(x))$$

Из этих равносильностей следует, что, если доказана теорема (1), то доказана и теорема (4), а если доказана теорема (2), то доказана и теорема (3).

10.6 Необходимые и достаточные условия

Некоторые теоремы существования сформулированы в виде « … для того , чтобы…, необходимо и достаточно, что …», или « … тогда и только тогда , когда …», а это конструкция $\forall x \in E(P(x) \to Q(x))$.

Рассмотрим теорему

$\forall x \in E(P(x) \to Q(x))$

Как отмечалось, множество истинности предиката $P(x) \to Q(x)$ есть множество $I_{\bar{P}} \cup I_Q$. Но тогда множеством ложности этого предиката будет $\overline{I_{\bar{P}} \cup I_Q} = I_P \cap I_{\bar{Q}}$. Последнее множество будет пустым лишь в случае, когда $I_P \subset I_Q$ (см. рисунок).

Итак, предикат $P(x) \to Q(x)$ является истинным для всех $x \in E$ том и в только в том случае, когда множество истинности предиката P(x) содержится в множестве истинности предиката Q(x). При этом говорят, что предикат Q(x) логически следует из предиката P(x), и предикат Q(x) называют необходимым условием для предиката P(x), а предикат P(x) – достаточным условием для Q(x).

Так, в теореме "Если x – число натуральное, то оно целое " предикат Q(x): " x – число целое " логически следует из предиката P(x): "x – число натуральное" , а предикат "x- число натуральное" является достаточным условием для предиката " x – целое число".

Часто встречается ситуация, при которой истинны взаимно обратные теоремы $\forall x \in E(P(x) \to Q(x))$ (1) $\forall x \in E(Q(x) \to P(x))$.(2)

Это, очевидно, возможно при условии, что $I_P = I_Q$.

В таком случае из теоремы (1)следует, что условия P(x)являются достаточными для Q(x), а из теоремы (2) следует, что условие P(x)является необходимым для Q(x).

Таким образом, если истинны теоремы (1) и (2), то условие P(x) является и необходимым, и достаточным для Q(x). Аналогично в этом случае условие Q(x)является необходимым и достаточным для P(x).

Иногда вместо логической связки "необходимо и достаточно " употребляют логическую связку "тогда и только тогда".

Так как здесь истинны высказывания (1) и (2), то истинно высказывание

$$\forall x \in E(P(x) \to Q(x)) \wedge \forall x \in E(Q(x) \to P(x)) \overset{35,19}{\equiv}$$
$$\equiv \forall x \in E(P(x) \leftrightarrow Q(x))$$

10.7 Доказательство теорем методом от противного

Доказательство теорем методом от противного обычно проводится по следующей схеме: предполагается, что теорема

$$\forall x \in E[P(x) \to Q(x)] \qquad (1)$$

не верна, т. е., существует такой объект x, что условие P(x) истинно, а заключение Q(x) – ложно. Если из этих предложений путем логических рассуждений приходят к противоречивому утверждению, то делают вывод о том, что исходное предположение неверно, и верна теорема (1).

Покажем, что такой подход дает доказательство истинности теоремы (1).

Действительно, предположение о том, что теорема (1) не справедлива , означает истинность ее отрицания, т. е. формулы $\overline{\forall x \in E[P(x) \to Q(x)]}$. Можно показать, что противоречивое утверждение, которое получается из допущенного предположения, как мы видели из ранее рассмотренных примеров, может быть записано как конъюнкция $C \wedge \overline{C} = A$, где C – некоторое высказывание.

Контрольные вопросы

1. Что такое «местность» предиката? Тождественно-истинный предикат? Тождественно-ложный предикат?
2. Как определяются логические операции над предикатами?
3. Чем отличается формула логики высказываний от формулы логики предикатов?
4. Какие формулы логики предикатов называют равносильными?

5. Дайте определение кванторов существования и всеобщности в случае одноместного и многоместного предиката.
6. Попробуйте доказать свойства кванторов.
7. Что такое терм? Функциональная сложность терма?
8. Что такое формула? Логическая сложность формулы?
9. Что такое предваренная нормальная форма?

Глава 11 Аксиомы и правила вывода исчисления предикатов

В исцелении предикатов используются формулы ЛИ, а понятие вывода определяется аналогично соответствующему понятию в ИВ. Аксиомами ИП являются все 4 группы аксиом ИВ и 5 группа :

V. 1. $\forall x F(x) \to F(y)$; аксиома Р1;

2. $F(y) \to \exists x F(x)$; аксиома Р2.

Правилами вывода ИП являются:

1) правила вывода ИВ (подстановка ПП и заключение(MP));

2) **правило обобщения (правило $\forall$ - введения)**:

Если $F \to G(x)$ - выводимая формула в ИП и F - не содержит переменной x, то формула $F \to \forall x\, G(x)$ также выводима в ИП

$$\frac{F \to G(x)}{F \to \forall x G(x)}$$ правило ПР1,

причем *G(x)* содержит свободные вхождения *x*, а F - не содержит;

3) **правило $\exists$ - введения**:

Если $G(x) \to F$ - выводимая формула в ИП и F - не содержит переменной x, то формула $\exists x\, G(x) \to F$ также выводима в ИП

причем *G(x)* содержит свободные вхождения *x*, а F - не содержит;

$$\frac{G(x) \to F}{\exists x G(x) \to F}$$ правило ПР2.

Других правил как и в ИВ довольно много, рассматривают по мере необходимости.

Пример Показать, что в ИВ из выводимости формулы *F(x)*, содержащей свободные вхождения x , не одно из которых не находится в области действия квантора по и *y*, следует выводимость формулы *F(y),* т.е. *F(x)* |- *F(y)*.

Доказательство

1. $F(x)$; ко условию
2. G ; любая предсказуемая формула
3. $F(x) \to G \to F(x))$; правило введения посылки
4. $G \to F(x)$; MP **(1, 3)**
5. $\dfrac{G(x) \to F}{\exists x G(x) \to F}$; ПР1 **(4)**
6. $\forall x F(x)$; MP **(2, 5)**
7. $\forall x F(x) \to F(y)$; аксиома P1
8. $F(y)$; MP **(6,7)**

Пример. Показать, что $\forall x F(x) \mid - \forall y F(y)$

Доказательство

1. $\forall x F(x)$; по условию
2. $\overline{x} \longrightarrow \overline{xy}$; аксиома P1
3. $\forall x F(x) \to \forall y F(y)$; правило ПР1 **(2)**
4. $\forall y F(y)$; MP (1,3)

Ниже представлено соответствие между высказываниями силлогистики и формулами ЛП.

Asp - "Всякое s есть p $" \leftrightarrow \forall x(s(x) \to p(x))$.

Esp - "Всякое s не есть p $" \leftrightarrow \forall x(s(x) \to \neg p(x))$.

Isp - "Некоторые s есть p $" \leftrightarrow \exists x(s(x) \to p(x))$.

Osp - "Некоторые s не есть p $" \leftrightarrow \exists x(s(x) \to \neg p(x))$.

Исследование выводимости 24 модусов, верных в силлогистике Аристотеля, в ИП привело к следующему результату. Если предполагать, что все классы сущностей не пуста, то приведенная выше замена силлогистических выражений формулами ЛП будет полностью справедлива. При допущении пустых классов сущностей оказываются невыводимыми все модусы силлогистики, в которых вывод носит частный характер, а обе посылки носят, общий характер.

Формализация процесса доказательства в логике предикатов

Доказательство демонстрирует, что некоторая формула ЛП является теоремой на заданном множестве аксиом Σ, т. е. результатом, логически выводимым из аксиом.

Положим, что $\Sigma = \{A_1, A_2, \ldots, A_n\}$, а теорема есть B. В таком случае можно утверждать, что доказательство показывает общезначимость формулы вида:

$A_1 \& A_2 \& \ldots \& An \rightarrow B$ (4.1)

Такое определение эквивалентно тому, что отрицание формулы (4.1.), т.е. формула $\neg (A_1 \& A_2 \& \ldots \& An \rightarrow B)$ является невыполнимой формулой. Ниже представлено поэтапное приведение формул ЛП к клаузальной форме, используемой в методе резолюций одном из наиболее эффективных методов доказательств в ЛП.

Следует отметить, что не существует алгоритма, позволявшего распознать общеназначимостъ, выполнимость или невыполнимость произвольной формулы ЛП.

Пример Даны два предиката: $B(x)$ = "x делится на 6"; $A(x)$ = "x делится на 3".Тогда $B(x) \supset A(x)$ = "Если x делится на 6, то x делится на 3" = И для всех x.Однако $B(x) \supset \forall x A(x)$ = "Если x делится на 6, то все x делятся на 3" не всегда истинно. Таким образом, применение правила П2 неправомерно, если B зависит от x.Если же к формуле $B(x) \supset A(x)$ применить правило П3, то получим $\exists x B(x) \supset A(x)$. После применения правила П2 получим $\exists x B(x) \supset \forall x A(x)$ = "Если некоторые x делится на 6, то все x делятся на 3" = Л. Таким образом, применение правила П3 также неправомерно, если B зависит от x.

11.1 Дополнительные правила вывода ИП

1. **Введение квантора общности**: $\dfrac{A(y)}{\forall x A(x)}$, где A(y) – результат правильной подстановки переменной y вместо x в A(x).

2. **Удаление квантора общности**: $\dfrac{\forall x A(x)}{A(y)}$, где A(y) – результат правильной подстановки терма y вместо x в A(x).

3. **Отрицание квантора общности**: $\dfrac{\neg \forall x A(x)}{\exists x \neg A(x)}$.

4. **Введение квантора существования**: $\dfrac{A(y)}{\exists x A(x)}$, где A(y) – результат правильной подстановки терма y вместо x в A(x).

5. **Удаление квантора существования**: $\dfrac{\exists y A(y)}{A(x)}$, где A(x) – результат правильной подстановки переменной x вместо y в A(y).

6. **Отрицание квантора существования**: $\dfrac{\neg \exists x A(x)}{\forall x \neg A(x)}$.

7. **Теорема дедукции.** Если Г – множество формул, A и B – формулы, и Г, A ④B. Тогда Г ④□A □□B.

Пример Обосновать правильность рассуждения, построив вывод.

а) Всякое нечетное натуральное число является разностью квадратов двух натуральных чисел. 5 – натуральное число. Следовательно, 5 – разность квадратов двух натуральных чисел

Пусть M – множество натуральных чисел. Введем предикаты:

A(x) = “x – нечетное число”.

B(x) – “x – разность квадратов двух чисел”.

Требуется построить вывод:

$\forall x(A(x) \square B(x)), A(5) \square \vdash B(5)$.

Построим вывод.

(1) $\forall x(A(x) \supset B(x))$ – гипотеза;

(2) $A(5)$ – гипотеза;

(3) $A(5) \supset B(5)$ – из (1) и удаления $\forall$;

(4) $B(5)$ – из (2) и (3) по *m. p.*

б) Все словари полезны. Все полезные книги высоко ценятся. Следовательно, все словари высоко ценятся.

Сначала формализуем наше рассуждение, введя следующие предикаты:

$A(x)$ = "x – словарь".

$B(x)$ = "x – полезен".

$C(x)$ = "x высоко ценится".

Требуется построить следующий вывод:

$\forall x(A(x) \supset B(x)), \forall x(B(x) \supset C(x)) \vdash \forall x(A(x) \supset C(x))$.

Построим этот вывод.

(1) $\forall x(A(x) \supset B(x))$ – гипотеза;

(2) $\forall x(B(x) \supset C(x))$ – гипотеза;

(3) $A(y) \supset B(y)$ – из (1) и удаления $\forall$;

(4) $B(y) \supset C(y)$ – из (2) и удаления $\forall$;

(5) $A(y) \supset C(y)$ – из (3) и (4) по правилу силлогизма;

(6) $\forall x(A(x) \supset C(x))$ – из (5) и введения $\forall$.

в) Всякий совершеннолетний человек, находящийся в здравом уме, допускается к голосованию. Джон не допущен к голосованию. Значит, он либо несовершеннолетний, либо не находится в здравом уме.

Формализуем наше рассуждение, введя следующие предикаты:

$A(x)$ = "x – совершеннолетний".

$B(x)$ = "x находится в здравом уме".

$C(x)$ = "x допущен к голосованию".

Введем константу d, обозначающую имя "Джон".

Требуется построить следующий вывод:

$\forall x(A(x)\&B(x) \supset C(x)), \neg C(d)) \vdash \neg A(d) \vee \neg B(d)$.

Построим этот вывод.

(1) $\forall x(A(x)\&B(x) \supset C(x))$ – гипотеза;

(2) $\neg C(d))$ – гипотеза;

(3) $A(d)\&B(d) \supset C(d)$ – из (1) и удаления $\forall$;

(4) $\neg C(d)) \supset \neg(A(d)\&B(d))$ – из (3) и правила контрапозиции;

(5) $\neg C(d)) \supset \neg A(d) \vee \neg B(d)$ – из (4) и отрицания конъюнкции;

(7) $\neg A(d) \vee \neg B(d)$ – из (2) и (5) по *т. р.*

(8)

11.2 Метод резолюций в ИП

Главная идея метода резолюций состоит в том, что, если одна и та же атомарная формула (или сопоставимые формулы) появляется в одном дизъюнкте без отрицания, а в другом - с отрицанием, то дизъюнкт, называемый резольвентой и получаемый в результате соединения этих двух дизъюнктов, из которых вычеркнута упоминавшаяся повторяющаяся формула (или сопоставимые формулы), является следствием указанных дизъюнктов.

Пусть S - множество дизъюнктов, исследуемых на невыполнимость. Алгоритм для метода резолюции представлен ниже.

Пока $\Lambda \in S$, искать l_1 , l_2 , S_1 , S_2 такие, что S_1 и S_2 есть дизъюнкты или факторы дизъюнктов;

$l_1 \in S_1$, $l_2 \in S_2$ и l_1, l_2 унифицируемы.

Вычислить резольвенту r , заменив S на S \{r}.

Пример. Доказать, что если G - группа, каждый элемент которой является обратным самому себе, то G - коммутативна.

Будем использовать следующие гипотезы:

H_1: $\forall x \, \forall y \, \forall z \, [(x \ y) \ z = x \ (y \ z)]$;

H_2: $\forall x \, [x \ e = x = e \ z]$;

H_3: $\forall x \, [x \ x = e]$

Заключение имеет следующий вид: C: $\forall x \, \forall y \, [x \times y = x \times y \,]$; Для исключения предиката равенства полагаем

$P(x, y, z) =_{def} x \ y = z$.

Представив гипотезы и отрицание заключения в клаузальной форме, получим множество дизъюнктов:

1. $\neg P(x, y, u) \vee P(y, z, \upsilon) \vee P(u, z, \omega) \vee P(x, \upsilon, \omega)$

2. $\neg P(x, y, u) \vee P(y, z, \upsilon) \vee P(x, \upsilon, \omega) \vee$

P(u, z, ω)

3. *P(x, e, x)*
4. *P(e, x, x)*
5. *P(x, x, x)*
6. *P(a, b, c)*
7. ┐ *P(b, a, c)*

Получение пустого дизъюнкта представлено ниже. В правой части списка указаны номера дизъюнктов, над которыми осуществляется резолюция, а также используемые унификаторы.

8. ┐ *P(x, z, υ)* v ┐ *P(e, z, ω)* v *P(x, υ , ω)*	*1 {(y, x) , (u, e)}, 5*
9. ┐ *P(b, z, υ)* v *P(a, υ , ω)* v *P(c, z, ω)*	*2 {(x, a) , (y, b), (u, c)}, 6*
10. ┐ *P(x, z, υ)* v *P(x, υ , z)*	*4 {(x, z)} ; 8 {(ω, z)}*
11. ┐ *P(a, e, ω)* v *P(c, b , ω)*	*5 {(x, b)} ; 9 {(z , b) , (υ, e)}*
12. *P(c, b, a)*	*3 {(x, a)} ; 11 {(ω, a)}*
13. *P(c, a, b)*	*10 {(x, c) , (z, b), (υ ,a)}, 12*
14. ┐ *P(x, y, u)* v ┐ *P(x, e, ω)* v *P(u, y , ω)*	*2 {(υ, e), (z, y)} , 5 {(ω, z)}*
15. ┐ *P(x, y, u)* v ┐ *P(u, y, x)*	*3, 14 {(ω, x)}*
16. *P(b, a, c)*	*13, 15 {(x, c) , (y, a), (u ,b)}*
17.	*7, 16*

Глава12 Неклассические логики

Из повседневной жизни известно, что основная масса понятий не является точно определенной. В 1989 году в США нечеткую логику даже хотели исключить из вузовских учебников, а теперь продуктами ее труда пользуются военные, финансисты, производственники и домохозяйки.

Впервые термин нечеткая логика (fuzzy logic) был введен амерканским профессором Лотфи Заде в 1965 году в работе "Нечеткие множества" в журнале "Информатика и управление".

Лотфи Аскер Заде (англ. Lotfi Asker Zadeh, 1921) — математик, основатель теории нечётких множеств и нечёткой логики, профессор Калифорнийского университета (Беркли). Родился в Баку, Азербайджан как Лотфи Алескерзаде (или Аскер Заде) от русской матери и отца азербайджанца иранского происхождения; с 1932 года жил в Иране, учился в Alborz High School и Тегеранском университете; с 1944 в Соединенных Штатах; с 1959 работает в Калифорнийском университете (Беркли).

В 1965 опубликовал основополагающую работу по теории нечётких множеств, в которой изложил математический аппарат теории нечётких множеств; в 1973 предложил теорию нечёткой логики; позднее — теорию мягких вычислений (soft computing); а также — теорию вербальных вычислений и представлений (computing with words and perceptions).

Основанием для создания новой теории послужил спор профессора со своим другом о том, чья из жен привлекательнее. К единому мнению они, естественно, так и не пришли. Это вынудило Заде сформировать концепцию, которая выражает нечеткие понятия типа "привлекательность" в числовой форме.

Одним из самых первых результатов применения этой теории стало создание управляющего микропроцессора на основе нечеткой логики, с помощью которого решалась задача управления зенитной ракетой, сбивающей межконтинентальные ракеты противника.

Однако, пожалуй, более впечатляющим выглядит применение нечеткой логики в дешевых изделиях массового

рынка - пылесосах, видеокамерах, микроволновых печах и т.п. Пионером в применении нечеткой логики в бытовых изделиях выступила фирма Matsuhita.

В феврале 1991 года она анонсировала первую <интеллектуальную> стиральную машину, в системе управления которой сочетались нечеткая логика. Автоматически определяя нечеткие входные факторы (объем и качество белья, уровень загрязненности, тип порошка и т.д.), стиральная машина безошибочно выбирала оптимальный режим стирки из 3800 возможных.

В классической логике существуют только две оценки. Если элемент принадлежит множеству, то 1, если не принадлежит, то 0. Например: есть множество шоколадных конфет. Они могут быть либо вкусными, либо нет. Это в обыкновенной логике. А предположим, что конфета - так себе, но невкусной ее все же не назовешь. Как быть? Человек говорит себе так: эта конфета скорее вкусная, чем нет. То есть делает предположение, высказывание. Нечеткая логика - это способ сгладить углы 0 и 1 и приблизить программирование компьютеров к мышлению человека.

Пример Прогноз погоды на любом из телевизионных каналов: завтра температура воздуха +5 градусов С, возможен дождь. В этом случае даже профессиональные синоптики не могут точно сказать будет дождь или нет. Это и есть проявление нечеткой логики: погода завтра может быть в данном случае как просто пасмурной, так и дождливой: события здесь предсказываются с некоторой долей уверенности (рангом).

В отличие от традиционной математики, требующей на каждом шаге моделирования точных и однозначных формулировок закономерностей, нечеткая логика предлагает совершенно иной уровень мышления, благодаря которому творческий процесс моделирования происходит на наивысшем уровне абстракции, при котором постулируется лишь минимальный набор закономерностей.

Недостатками нечетких систем являются:

- *отсутствие стандартной методики конструирования нечетких систем;*

- *невозможность математического анализа нечетких систем существующими методами;*
- *применение нечеткого подхода по сравнению с вероятностным не приводит к повышению точности вычислений.*

Как видно, классические методы управления хорошо работают при полностью детерминированном объекте управления и детерминированной среде, а для систем с неполной информацией и высокой сложностью объекта управления оптимальными являются нечеткие методы управления. В правом верхнем углу рисунка приведена еще одна современная технология управления - с применением искусственных нейронных сетей.

12.1 Базовые понятия нечеткой логики

Пример Нечеткое множество для термина молодой.

До 16 лет нельзя однозначно утверждать, что человек молодой (например, 15-летие относится к термину

молодой с рангом около 0,9). Зато диапазону от 16 до 30 лет можно смело присвоить ранг 1, т.е. человек в этом возрасте молодой. После 30 лет человек вроде уже не молодой, но еще и не старый, здесь принадлежность (ранг) термина молодой возрасту будет принимать значения в интервале от 0 до 1. И чем больше возраст человека, тем меньше становится его принадлежность к молодым, т.е. ранг будет стремиться к 0.

Рассуждая таким образом, было получено нечеткое множество, описывающее понятие молодости для всего диапазона возрастов человека. Если ввести остальные термины (например, очень молодой, старый и т.д.), то можно охарактеризовать такую переменную как возраст, состоящую из нескольких нечетких множеств и полностью перекрывающую весь жизненный период.

12.2 Основные операции с нечеткими множествами

1. **Включение.** Говорят, что A содержится в B, если для всех "xЄ E выполняется

$$\mu A(x) \subset \mu B(x)$$

Обозначение: $A \subset B$. Иногда используют термин *доминирование*, т. е. в случае, когда $A \subset B$, говорят, что B доминирует A.

2. **Равенство.** Нечеткие множества A и B равны, если $\forall x \in E$ $\mu A(x) = \mu B(x)$. Обозначение: A = B.

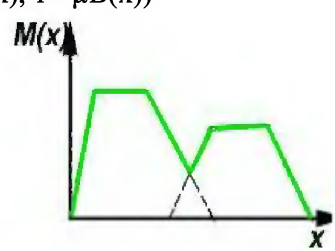
3. **Разность.** $A - B = A \cap B'$ с функцией принадлежности:

$$\mu A - B\ (x) = \min (\mu A(x), 1 - \mu B(x))$$

4.**Объединение**

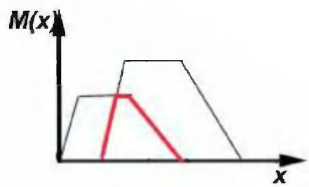
$A \dot{\cup} B$ - наименьшее нечеткое подмножество, включающее как A так и B, с функцией принадлежности:

$\mu A \dot{\cup} B\ (x) = \max (\mu A(x)\ \mu B(x))$



5. Пересечение

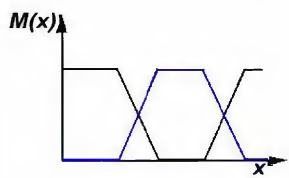
A∩B - наибольшее нечеткое подмножество, содержащееся одновременно в A и B: $\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$



3.*Дополнение* A и B дополняют друг друга, если ∀ xЄ E

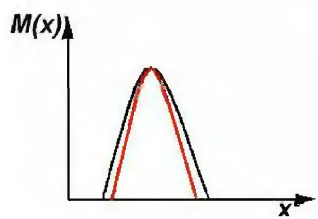
$$\mu A(x) = 1 - \mu B(x)$$

Дополнение соответствует логическому отрицанию.



4.Концентрация

$$kon_n A = \{x, \quad \mu_A(x)^n\}$$



Пример. Пусть A = 0,4/x1 + 0,2/ x2 + 0/x3 + 1/x4;
B = 0,7/ x1 + 0,9/ x2 + 0,1/ x3 + 1/ x4 ,
C = 0,1/ x1 + 1/ x2 + 0,2/ x3 + 0,9/ x4
Тогда:

1) A⊂ B, т. е. A содержится в B ;

C несравнимо ни с A, ни с B, т.е. пары {A,C} и {B,C} - пары не доминируемых нечетких множеств.

2) A≠ B ≠ C

3) A'=0,6/ x_1+0,8/x_2+ 1/x_3+ 0/ x_4; = 0,3/ x_1+0,1/x_2+0,9/ x_3+0/ x_4

4) A∩B = 0,4/ x_1 + 0,2/ x_2 + 0/ x_3 +1/ x_4

5) AÙB = 0,7/ x_1 + 0,9/ x_2 + 0,1/ x_3 + 1/x_4

6)A - B =0,3/ x_1+0,1/ x_2+0/ x_3 + 0/x_4; - =0,6/ x_1+0,8/ x_2 + 0,1/ x_3 + 0/ x_4

Для нечетких множеств для операций пересечения ∩ и объединения Ù выполняются свойства коммутативности, ассоциативности, идемпотентности, дистрибутивности и теоремы де Моргана.

Кроме того, AÙØ = A, где Ø - пустое множество,

т.е.,$\mu_{Ø}$ (x),∀ xЄE

A∩Ø =Ø; A∩= A ; AÙE = E.

В отличие от четких множеств, для нечетких множеств в общем случае:

A∩A'≠Ø , AÙA'≠E

✓ **Замечание.** Введенные выше операции над нечеткими множествами основаны на использовании операций max и min.

Все системы с нечеткой логикой функционируют по одному принципу:

показания измерительных приборов фаззифицируются (переводятся в нечеткий формат), обрабатываются , дефаззифицируются и в виде привычных сигналов подаются на исполнительные устройства.

➢ **Определение Фаззификация** - сопоставление множества значений x ее функции принадлежности M(x), т.е. перевод значений x в нечеткий формат (пример с термином молодой). Дефаззификация - процесс, обратный фаззификации.

Фаззификация (переход к нечеткости). Точные значения входных переменных преобразуются в значения лингвистических переменных посредством применения некоторых положений теории нечетких множеств, а именно - при помощи определенных функций принадлежности.

Степень принадлежности - это не вероятность, т.к. неизвестна функция распределения, нет повторяемости экспериментов.

Пример Прогноз погоды - два взаимоисключающих события: будет дождь и не будет.

Присвоить им некоторые ранги, то сумма этих рангов необязательно будет равна 1, но если равенство все-таки есть, то

нечеткое множество считается нормированным. Значения функции принадлежности M(x) могут быть взяты только из априорных знаний, интуиции (опыта), опроса экспертов.

12.3 Понятие лингвистической переменной

➢ **Определение Лингвистическая переменная** - переменная, значениями которой являются не числа, а слова естественного языка, называемые термами.

Лингвистической переменной (ЛП) называется пятерка (ß, T, X, G, M), где ß - наименование лингвистической переменной; T - множество ее значений (терм-множество), представляющих собой наименования нечетких переменных, областью определения каждой из которых является множество X.

Множество T называется **базовым терм-множеством** лингвистической переменной;

G - синтаксическая процедура, позволяющая оперировать элементами терм-множества T, в частности, генерировать новые термы (значения).

Множество TÙG(T), где G(T) - множество сгенерированных термов, называется **расширенным терм-множеством** лингвистической переменной;

M - семантическая процедура, позволяющая превратить каждое новое значение лингвистической переменной, образуемое процедурой G, в нечеткую переменную, т.е. сформировать соответствующее нечеткое множество.

С термином «лингвистическая переменная» можно связать любую физическую величину, для которой нужно иметь больше значений, чем только ДА и НЕТ. В этом случае вы определяете необходимое число термов и каждому из них ставите в соответствие некоторое значение описываемой физической величины.

В настоящее время сложилось мнение, что для большинства приложений достаточно 3-7 термов на каждую переменную. Минимальное значение числа термов вполне оправданно. Такое определение содержит два экстремальных значения (минимальное и максимальное) и среднее. Для большинства применений этого вполне достаточно.

Что касается максимального количества термов, то оно не ограничено и зависит целиком от приложения и требуемой точности описания системы. Число же 7 обусловлено емкостью кратковременной памяти человека, в которой, по современным представлениям, может храниться до семи единиц информации.

Пример В случае управления мобильным роботом можно ввести две лингвистические переменные: ДИСТАНЦИЯ (расстояние до помехи) и НАПРАВЛЕНИЕ (угол между продольной осью робота и направлением на помеху).

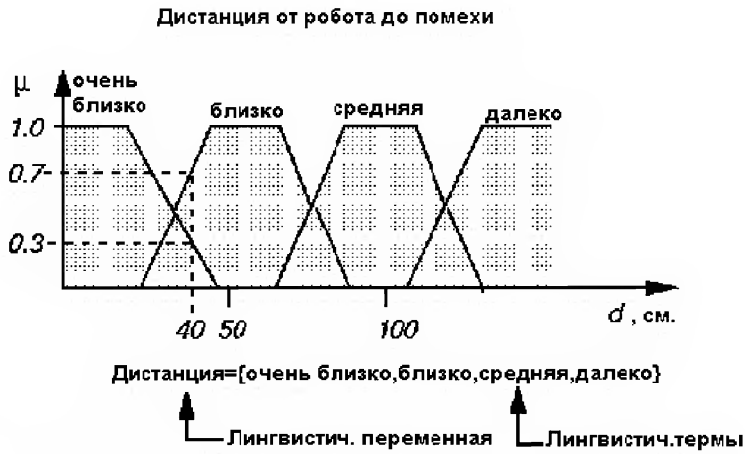
Рассмотрим лингвистическую переменную ДИСТАНЦИЯ. Значениями ее можно определить термы ДАЛЕКО, СРЕДНЯЯ, БЛИЗКО и ОЧЕНЬ БЛИЗКО.

Для физической реализации лингвистической переменной необходимо определить точные физические значения термов этой переменной.

Пусть переменная ДИСТАНЦИЯ может принимать любое значение из диапазона от нуля до бесконечности.

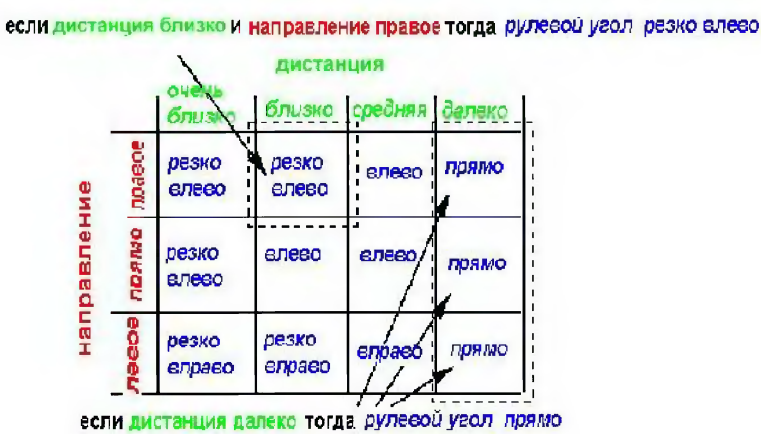
В таком случае каждому значению расстояния из указанного диапазона может быть поставлено в соответствие некоторое число от нуля до единицы, которое определяет степень принадлежности данного физического расстояния (допустим 40 см) к тому или иному терму лингвистической переменной ДИСТАНЦИЯ

Пример В нашем случае расстоянию 40 см. можно задать степень принадлежности к терму ОЧЕНЬ БЛИЗКО равную 0,7 , а к терму БЛИЗКО– 0,3 (см. рис.1.). Конкретное определение степени принадлежности может проходить только при работе с экспертами.



Пример Переменной НАПРАВЛЕНИЕ, которая может принимать значения в диапазоне от 0 до 360 градусов, зададим термы ЛЕВОЕ, ПРЯМО И ПРАВОЕ.

Теперь необходимо задать выходные переменные. В рассматриваемом примере достаточно одной, которая будет называться РУЛЕВОЙ УГОЛ. Она может содержать термы: РЕЗКО ВЛЕВО, ВЛЕВО, ПРЯМО, ВПРАВО, РЕЗКО ВПРАВО. Связь между входом и выходом запоминается в таблице нечетких правил:



Каждая запись в данной таблице соответствует своему нечеткому правилу, например:

Если ДИСТАНЦИЯ БЛИЗКО и НАПРАВЛЕНИЕ ПРАВОЕ, тогда РУЛЕВОЙ УГОЛ РЕЗКО ВЛЕВО

Таким образом, мобильный робот с нечеткой логикой будет работать по следующему принципу:

данные с сенсоров о расстоянии до помехи и направлении на нее будут фаззифицированы, обработаны согласно табличным правилам, дефаззифицированы и полученные данные в виде управляющих сигналов поступят на привода робота.

12.4 Нечеткие отношения

Нечеткое n-арное отношение определяется как нечеткое подмножество R на E, принимающее свои значения в M.

В случае n = 2 и M = [0, 1] нечетким отношением R между множествами X = E_1 и У = E_2 будет называться функция R: (X, У)- >[O, 1], которая ставит в соответствие каждой паре элементов (x,y)ЄX×У величину $\mu_R(x,y)$Є[0,1]

Нечеткое отношение на X×У обычно пишут в виде

xЄX, yЄ У : xRy.

Пример Пусть X = { x_1, x_2, x3}, У = {y_1, y_2, y_3, y_4}, M = [0, 1]. Нечеткое отношение R = XRY может быть задано следующей таблицей.

	y_1	y_2	y_3	y_4
x_1	0,2	0,3	0,1	0,5
x_2	1	0,7	0	0,4
x_3	0,7	0	1	0,6

12.5 Нечеткие выводы

Используемый в экспертных и управляющих системах механизм нечетких выводов в своей основе имеет базу знаний,

формируемую специалистами-экспертами предметной области в виде совокупности нечетких предикатных правил вида:

Правило 1: если x есть A_1, тогда y есть B_1

Правило 2: если x есть A_2, тогда y есть B_2.

Правило N: если x есть A_n, тогда y есть B_n где x - входная переменная, y - переменная вывода (имя для значения данных, которое будет вычислено); A и B - функции принадлежности, определенные соответственно на x и y.

В общем случае **нечеткий логический вывод осуществляется в следующие этапы**

1. Нечеткость (введение нечеткости, фазификация, fuzzification).

Функции принадлежности, определенные на входных переменных, применяются к их фактическим значениям, для того чтобы определить степени истинности каждой предпосылки каждого правила.

2. Логический вывод.

Вычисленное значение истинности для предпосылок каждого правила применяется к заключениям каждого правила. Это приводит к одному нечеткому подмножеству, которое будет назначено каждой переменной вывода для каждого правила. В качестве правил логического вывода обычно используются только операции min (минимум) или prod (умножение).

При операции min функция принадлежности вывода "отсекается" по высоте, соответствующей вычисленной степени истинности предпосылки правила (нечеткая логика "И").

При операции умножения функция принадлежности вывода масштабируется при помощи вычисленной степени истинности предпосылки правила.

3. Композиция.

Все нечеткие подмножества, назначенные к каждой переменной вывода (во всех правилах), объединяются вместе, чтобы сформировать одно нечеткое подмножество для каждой переменной вывода. При подобном объединении обычно используются операции max (максимум) или sum (сумма).

При операции max комбинированный вывод нечеткого подмножества конструируется как поточечный максимум по

всем нечетким подмножествам (нечеткая логики "ИЛИ"). В случае операции суммы комбинированный вывод нечеткого подмножества конструируется как поточечная сумма по всем нечетким подмножествам, назначенным переменной вывода правилами логического вывода.

4.Дополнительно

Может быть введен этап приведения к четкости (дефазификация, defuzzification), используемый, когда целесообразно преобразовать нечеткий набор выводов в четкое число.

12.6 Функции принадлежности

➢ **Определение Степень принадлежности** определяется так называемой функцией принадлежности **M(d),** где d-расстояние до помехи.

$$A=\{\mu_A(x)/x\},$$

где $\mu_A(x)$ - *характеристическая функция принадлежности* (или просто функция принадлежности), принимающая значения в некотором вполне упорядоченном множестве M.

Функция принадлежности указывает степень (уровень) принадлежности элемента x подмножеству A. Множество M называют множеством принадлежностей.

Если M = {O, 1}, то нечеткое подмножество A может рассматриваться как обычное (четкое) множество.

Вид функции принадлежности может быть абсолютно произвольным. Сейчас сформировалось понятие о так называемых стандартных функциях принадлежности.

Z - функция П - функция Л - функция S - функция

Стандартные функции принадлежности легко применимы к решению большинства задач.

12.7 Основные характеристики нечетких множеств

1. Величина $\mu_A(x)$ называется **высотой** нечеткого множества A.

Нечеткое множество A **нормально**, если его высота равна 1, т.е. верхняя граница его функции принадлежности ($\mu_A(x_1) = 1$).

При sup $\mu_A(x) < 1$ нечеткое множество называется **субнормальным**.

2. Нечеткое множество **пусто**, если$\forall$ xЄ E $\mu_A(x)=0$.

3. Нечеткое множество **унимодально**, если $\mu_A(x) = 1$ только на одном x из E

4. **Носителем** нечеткого множества A является обычное подмножество со свойством $\mu_A((x) > 0$, т.е. носитель A = {x/xЄ E, $\mu_A((x) > 0$}.

5. Элементы xЄ E , для которых $\mu_A(x)= 0,5$, называются **точками перехода** множества A.

- **Пример** Пусть E = {0, 1,2, ..., 10}, M = [0, 1].

Нечеткое множество "Несколько" можно определить следующим образом: "Несколько" = 0,5/3+0,8/4 + 1/5 + 1/6+0,8/7 + 0,5/8;

его характеристики: высота = 1,

носитель - {3, 4, 5, 6, 7, 8},

точки перехода - {3, 8}.

- **Пример** Пусть E = {0, 1, 2, 3, . . ., n, . . .}.

Нечеткое множество "Малый" можно определить:

"Малый" = { $\mu_{Малый}(n) = ((1) / (1 + (n. /10))^2) /n$)}

- **Пример** Пусть E - множество целых чисел: E={-8, -5, -3, 0, 1, 2, 4, 6, 9}. Тогда нечеткое подмножество чисел, по абсолютной величине близких к нулю, можно определить, например, так:

A = {0/-8+0,5/-5+0,6/-3 + 1/0+0,9/1 + 0,8/2 + 0,6/4+0,3/6+0/9}.

12.8 Алгоритм формализации задачи в терминах нечеткой логики

Шаг 1. Для каждого терма взятой лингвистической переменной найти числовое значение или диапазон значений, наилучшим образом характеризующих данный терм.

Шаг 2. После определения значений с единичной принадлежностью необходимо определить значение параметра с принадлежностью «0» к данному терму.

Шаг 3. Для определения промежуточных значений выбираются П- или Л-функции из числа стандартных функций принадлежности. Для значений, соответствующих экстремальным значениям параметра, выбираются S- или Z-функции принадлежности.

12.9 Разработка нечетких правил

На этом этапе определяются правила, связывающие лингвистические переменные.

Типичное продукционное правило состоит из антецедента (часть ЕСЛИ …) и консеквента (часть ТО …). Антецедент может содержать более одной посылки. В этом случае они объединяются посредством логических связок И или ИЛИ.

Процесс вычисления нечеткого правила называется нечетким логическим выводом и подразделяется на два этапа: обобщение и заключение.

На первом шаге логического вывода необходимо определить степень принадлежности всего антецедента правила. Для этого в нечеткой логике существуют два оператора: MIN(…) и MAX(…). Первый вычисляет минимальное значение степени принадлежности, а второй - максимальное значение. Когда применять тот или иной оператор, зависит от того, какой связкой соединены посылки в правиле. Если использована связка И, применяется оператор MIN(…). Если же посылки объединены связкой ИЛИ, необходимо применить оператор MAX(…). Ну а если в правиле всего одна посылка, операторы вовсе не нужны.

Чтобы исполнительное устройство смогло отработать полученную команду, необходим этап управления, на котором мы избавляемся от нечеткости и который называется дефаззификацией.

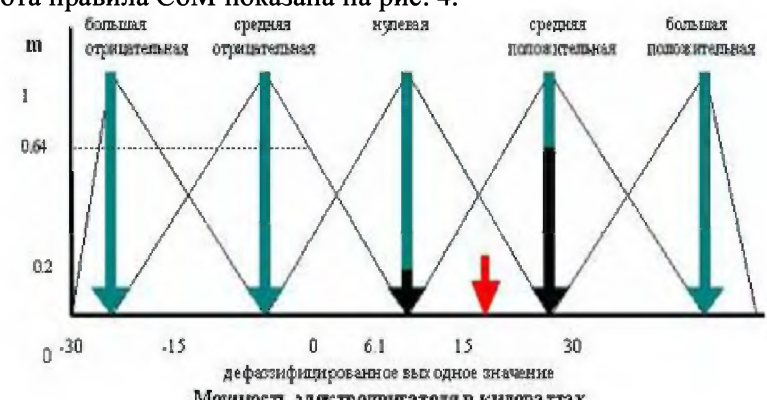
Дефаззификация (устранение нечеткости)

На этом этапе осуществляется переход от нечетких значений величин к определенным физическим параметрам, которые могут служить командами исполнительному устройству.Результат нечеткого вывода, конечно же, будет нечетким.

Для устранения нечеткости окончательного результата существует несколько методов. Рассмотрим некоторые из них.

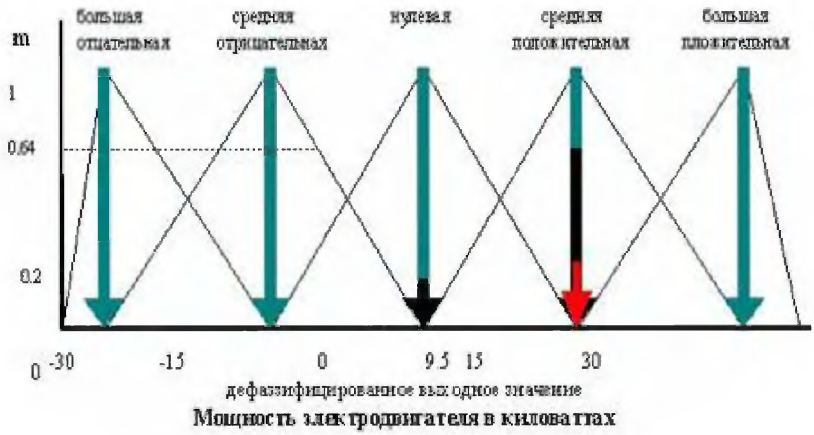
12.10 Метод центра максимума (CoM)

Так как результатом нечеткого логического вывода может быть несколько термов выходной переменной, то правило дефаззификации должно определить, какой из термов выбрать. Работа правила CoM показана на рис. 4.



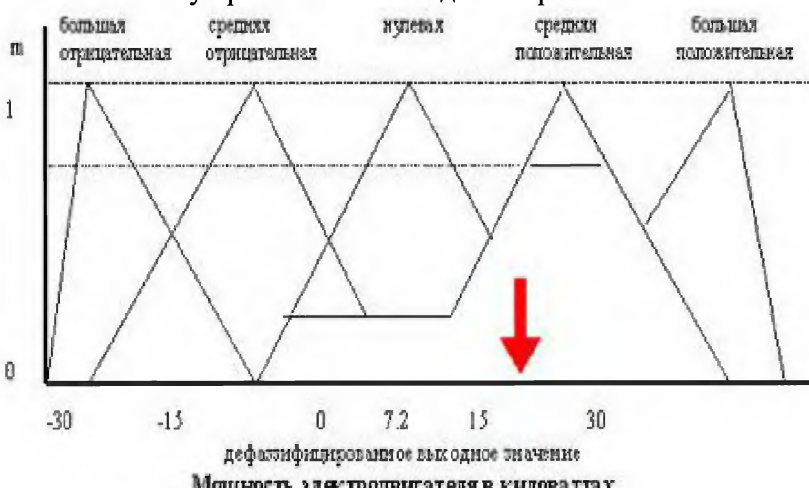
12.11 Метод наибольшего значения (MoM)

При использовании этого метода правило дефаззификации выбирает максимальное из полученных значений выходной переменной. Работа метода ясна из рис. 5.

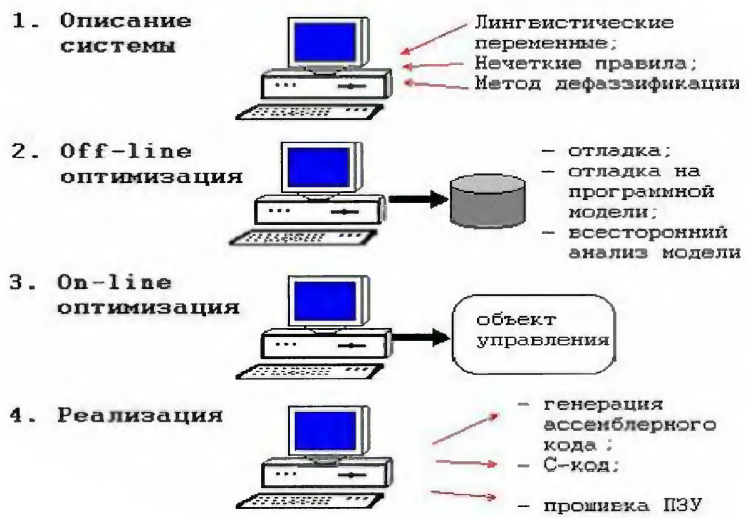


12.12 Метод центроида (CoA)

В этом методе окончательное значение определяется как проекция центра тяжести фигуры, ограниченной функциями принадлежности выходной переменной с допустимыми значениями. Работу правила можно видеть на рис. 6.



Процесс разработки проекта нечеткой системы управления на fuzzy TECH разбивается, как уже говорилось, на четыре основных этапа. Все они схематично показаны на рис. 7.



Описание системы

На этом этапе при помощи средств, доступных в fuzzy TECH, задача формализуется. Здесь необходимо описать лингвистические переменные, которые вы будете использовать; их функции принадлежности; описать стратегию управления посредством нечетких правил, которые вы сможете объединить в единую базу правил или знаний о системе. В целом CASE-технология, на основе которой построен пакет, позволяет все эти действия выполнить только посредством общения с экраном ЭВМ, не заглядывая в программный код.

Глава 13 Многозначные логики

Двузначная логика предполагает истинность и ложность высказываний («0» или «1»). В многозначных логиках число значений истинности аргументов и функций может быть даже (в общем случае) бесконечным.

Обозначим

Nx или ¬**x**-отрицание,

Cxy или **x** ⊃ **y** – импликацию,

Kxy или **x** ∧y – конъюнкцию,

Axy или **x** ∨ **y** – дизъюнкцию. З

Значение функции от аргумента а обозначим как **[a].**

Развитие многозначных логик приводит к тому, что ряд утверждений, являющихся тождественно-истинными в одной логической системе, становятся нетождественно-истинными в другой системе.

Рассмотрим несколько примеров многозначных логик, которые подтверждают это утверждение.

13.1 Трехзначная система Я. Лукасевича

Трехзначная система или пропозиционная логика была построена Я. Лукасевичем в 1920 году.

Лукасевич (Łukasiewicz) Ян (21.12.1878, Львов, — 13.11.1956, Дублин), польский логик, член Польской АН (1937), профессор университетов во Львове (1906—15) и Варшаве (1915—39. Работал в области логических проблем индукции и причинности и логических оснований теории вероятностей.

Построил первую систему многозначной логики, а с её помощью — систему модальной логики. Разработал оригинальный язык для формализации логических и математических выражений (так называемая бесскобочная символика Л.).

Лукасевич обозначил **«истину» за «1»,**

«ложь» за «0» и

ввел третье значение – «нейтрально» - ½.

Основными функциями им были взяты

отрицание и импликация,

а производными – **конъюнкция и дизъюнкция**.

Тавтология в логике Я. Лукасевича принимает значение «1».

Отрицание и импликация определяются таблицами 1, 2 и равенствами:

Таблица 1

X	Nx
1	0
½	½
0	1

Таблица 2

x ⊃ y	1	½	0
1	1	½	0
½	1	1	½
0	1	1	1

[Nx] = 1 – [x];

[Cxy] = 1, если [x] ≤ [y];

[Cxy] = 1 – [x] + [y], если [x] > [y] или в общем виде;

[Cxy] = min(1,1 – [x] + [y]).

Конъюнкция в системе Лукасевича определяется как минимум значений аргументов: **[Kxy] = min([x], [y]),**

Дизъюнкция – как максимум значений аргументов x и y:

[Axy] = max([x], [y]).

Очевидно, что в этих определениях **не являются законами** логики (тавтологии) законы двузначной логики: исключенного третьего, непротиворечия, отрицания законов непротиворечия и исключенного третьего.

Поэтому система Лукасевича не является отрицанием двузначной логики.

В его логике правило снятия двойного отрицания, четыре правила де Моргана и правило контрапозиции: $\neg a \supset b \sim b \supset a$ являются тавтологиями. Не являются тавтологиями правила приведения к абсурду двузначной логики:

$(\neg x \supset \neg x) \supset x$ и

$(x \supset (y \wedge \neg y)) \supset \neg x$

(т.е. если из x вытекает противоречие, то из этого следует отрицание x).

Доказывается это, если задать [x] = ½ и [y] = ½.

В данной логике не являются тавтологиями и ряд формул, выражающие правильные дедуктивные умозаключения традиционной логики, формализованные средствами алгебры логики.

В соответствии с одной из ее интерпретаций, высказывания должны делиться не просто на истинные и ложные, а **на истинные, ложные и парадоксальные**

Значение "парадоксально" приписывается высказываниям типа "Данное утверждение является ложным", т. е. тем высказываниям, из допущения истинности которых вытекает их ложность, а их допущения ложности – истинность

К бессмысленным относятся высказывания типа "Наполеон - наибольшее натуральное число" и т. п. Это значение истолковывалось и как "неизвестно" или "неопределенно". Неопределенное высказывание - это высказывание, относительно которого в силу к.-л. (возможно, меняющихся от случая к случаю) оснований нельзя сказать, что оно истинно или ложно.

К неопределенным могут относиться, в частности, высказывания, истинностное значение которых является разным в разные моменты времени ("Идет дождь"), высказывания с различного рода переменными и т. д.

1.**Константы**, т.е. функции, для которых все аргументы являются фиктивными. В трехзначной логике имеется три константных функции

$\mathbf{f_0 = 0,}$

$\mathbf{f_1 = 1,}$

$\mathbf{f_2 = 2.}$

Отметим, что “0” здесь соответствует значению истинности “ложь”, “1” – значению “неопределенно, неизвестно”, “2” – значению “истина”.

2.Наиболее важными функциями одной переменной являются **характеристические функции,** число которых равно числу значений истинности логики, в данном случае – трем.

Характеристическая функция $\varphi_i(x)$, называемая характеристической функцией i-го порядка, определяется следующим образом:

$$\varphi_i(x) = \begin{cases} 2, & \text{если} \quad x = i \\ 0, & \text{если} \quad x \neq i \end{cases} \qquad (1)$$

Таблица 1. Характеристические функции

x	$\varphi_0(x)$	$\varphi_1(x)$	$\varphi_2(x)$
0	2	0	0
1	0	2	0
2	0	0	2

3.**Обобщенная характеристическая функция** e_{ij}, задаваемая следующим образом:

$$e_{ij}(x) = \begin{cases} j, & \text{если} \quad x = i \\ 0, & \text{если} \quad x \neq i \end{cases} \qquad (2)$$

Таблица 2. Обобщенные характеристические функции.

x	$e_{ij}(x)$								
	e_{00}	e_{10}	e_{20}	e_{01}	e_{11}	e_{21}	e_{02}	e_{12}	e_{22}
0	0	0	0	1	0	0	2	0	0
1	0	0	0	0	1	0	0	2	0
2	0	0	0	0	0	1	0	0	2

4.Важной является **функция инверсии**, служащая обобщением функции отрицания:

$$\neg x = \bar{x} = 2 - x \qquad (3)$$

Таблица 3. Функция инверсии.

x	$\neg x$
0	2
1	1
2	0

5.Функция **циклического отрицания**:

$$\hat{} x = \hat{x} = \vec{x} = x \oplus 1 (\mathrm{mod}\ 3) \qquad (4)$$

Таблица 4. Функция циклического отрицания.

x	^x
0	1
1	2
2	0

6.Среди функций двух переменных особо важную роль играют **функции трехзначной дизъюнкции и трехзначной конъюнкции**. Эти функции определяются на основании соотношений:

$a \vee b = \max(a, b)$; **(5)**

$a \wedge b = a \wedge b = \min(a, b)$. **(6)**

Таблица 5. Трехзначные дизъюнкция и конъюнкция.

a	b	$a \vee b$	$a \wedge b$
0	0	0	0
0	1	1	0
0	2	2	0
1	0	1	0
1	1	1	1
1	2	2	1
2	0	2	0
2	1	2	1
2	2	2	2

7.Важными функциями трехзначной логики являются **функция сложения по модулю три** $a + b \pmod 3 = a \oplus b$ и **функция умножения по модулю три** $a * b \pmod 3 = a \otimes b$ без учета переносов. Кроме того, представляет особый интерес трехзначная **функция Вебба**, которая определяется с помощью следующего соотношения:

a | b = max (a, b) + 1(mod 3) = (a ∨ b) ⊕ 1 (7)

Таблица 6. Функции сложения и умножения по mod 3 и функция Вебба.

a	b	a ⊕ b	a ⊗ b	a | b
0	0	0	0	1
0	1	1	0	2
0	2	2	0	0
1	0	1	0	2
1	1	2	1	2
1	2	0	2	0
2	0	2	0	0
2	1	0	2	0
2	2	1	1	0

С помощью перечисленных выше функций можно представить любые трехзначные функции алгебры логики. Для представления функций в многозначной логике и для синтеза схем ограничиваются рассмотрением только таких базисов и полных систем, которые оказались удобными для этой цели.

Важнейшие и наиболее интересные с точки зрения практики системы такого типа следующие:

1) **Система Поста**. Постом было показано, что в любой многозначной логике полна система, состоящая из дизъюнкции и цикла, т. е. любую троичную функцию можно выразить через дизъюнкцию и циклическое отрицание.

2) **Система Россера и Тьюкетта**. Полную систему функций в многозначной логике составляют характеристические функции, функции конъюнкции, дизъюнкции, функции константы.

3) **Система Вебба**. Полную систему составляет для любой многозначной логики функция Вебба.

4) **Модульная логика** (или модулярная). Если k – простое число, то функции сложения по модулю k и умножения по модулю k образуют в k-значной логике полную систему.

Кроме того, любая функция многозначной логики может быть представлена в форме дизъюнкций характеристических конъюнкций, которая называется

многозначной дизъюнктивной совершенной нормальной формой (МДСНФ),

и в форме конъюнкций характеристических дизъюнкций , которая называется

многозначной конъюнктивной совершенной нормальной формой (МКСНФ).

Функции конъюнкции и дизъюнкции в многозначной логике имеют свойства, аналогичные свойствам двузначных функций конъюнкции и дизъюнкции. В частности, с помощью инверсии они связаны между собой известными формулами де Моргана.

В полных системах Поста, Вебба, и модульной системе аналитическое выражение функций трехзначной логики получается довольно громоздким и менее прозрачным, чем в системе Россера, Тьюкетта или при представлении функции в виде ТДСНФ или ТКСНФ.

13.2 Логика Гейтинга

Из закона исключенного терьего в двузначной логике выводятся:

1. $\neg\neg x \supset x$ **2.** $x \supset \neg\neg x$

Гейтинг создал трехзначную пропозициональную логику, основываясь на утверждении, что истинным является лишь $x \supset \neg\neg x$.

Импликация и отрицание (таблицы 3, 4 отличаются от определений этих операций.

В предыдущей логике лишь в одном случае “истина” обозначена Гейтингом за “1”, “ложь” – “0” и введено понятие “неопределенность” - ½. Тавтология принимает значение 1.

Таблица 13

X	Nx
1	0
½	0
0	1

Таблица 14

x \ y	1	½	0
1	1	½	0

½ 0	1 1	1 1	0 1

[Cxy] = 1, если [x] ≤ [y];

[Cxy] = [y], если [x] > [y]

Кху и Аху определены как минимум и максимум значений аргумента.

Очевидно, что учет лишь значений функций 1 и 0 приводит к вычислению матрицы двузначной логики.

В логике Гейтинга законы непротиворечия, формула

(x ⊃ y) ⊃ (y ⊃ x), де Моргана и исключенного четвертого: (¬x ∨ x∨ ¬¬x) являются тавтологиями.

Но ни закон исключенного третьего, ни его отрицание не являются тавтологиями.

13.3 Трехзначная система Бочвара Д.А.

Система создавалась Бочваром Д.А. для разрешения парадоксов классической математической логики методом формального доказательства бессмысленности определенных высказываний.

Например, ему удалось разрешить парадокс Рассела о множестве всех нормальных множеств и доказать несуществование такого предмета, как множество всех нормальных множеств. Это означает, что множество всех нормальных множеств нельзя рассматривать как фиксированный объект, не изменяющийся от времени.

Создавая свою систему Д.А. Бочвар, разделил высказывания на имеющие смысл («истина» или «ложь») и бессмысленными. Он выделил внешние и внутренние формы (функции).Внутренние функции называются классическими содержательными функциями переменных высказываний, а внешние – неклассическими.

Обозначив «истина» за R или 1, «ложь» – F или 3, «бессмысленность» – S или 2, автор ввел отрицание внутреннее – «~a», внешнее отрицание – «┐a», «ā» – внутреннее отрицание внешнего утверждения, «≡» – внешняя равнозначность, «↔» – внешняя равносильность.

В логике Бочвара Д.А. законы тождества, отрицания двузначной логики не являются тавтологиями. Отрицание закона тождества как раз и позволило разрешить парадокс Рассела. В логике Бочвара Д.А. формулы, приведенные ниже являются противоречиями: $a \wedge \neg a$; $a \leftrightarrow \bar{a}$; $a \equiv \neg a$.

13.4 К - значная логика Поста Е.Л.

Логика Поста является обобщением частного случая – двузначной логики, когда К=2.

Пост (Post) Эмиль Леон (11.2.1897, Августов, Польша, — 21.4,1954, Нью-Йорк), американский математик и логик. Читал лекции по математике и логике в Колумбийском, Нью-Иоркском и др. университетах США. Им получен ряд фундаментальных результатов в математической *логике;* одно из наиболее употребительных определений понятий *непротиворечивости* и полноты формальных систем (*исчислений*); доказательства функциональной *полноты* и дедуктивной полноты (в широком и узком смысле) исчисления высказываний; изучение систем *многозначной логики* с более чем 3 значениями истинности; одно из первых (независимое от А. М. *Тьюринга*) определении понятия *алгоритма* в терминах «абстрактной вычислительной машины» и формулировка основного тезиса теории алгоритмов о возможности описать любой конкретный алгоритм посредством этого определения; результаты о выразимости общерекурсивных функций и предикатов через примитивно рекурсивные, в частности т. н. теорема о нормальной форме; первые (одновременно с А. А. *Марковым*) доказательства алгоритмической неразрешимости ряда проблем математической логики и алгебры и др.

Действительно, по Посту значения истинности принимают значения 1, 2,…,К (при $К \geq 2$ и К – конечно).

В этих терминах формула является тавтологией, когда принимает такое значение i, что $1 \leq i \leq S$, где $1 \leq S \leq К\text{-}1$. Значения 1,…, S называются *выделенными* (отмеченными). При этом S может быть и больше 2.

Пост ввел N^1_x – циклическое отрицание, N^2_x – симметричное отрицание. Они определяются таблицей 5 и равенствами.

Таблица5

X	N^1_x	N^2_x
1	2	K
2	3	K-1
3	4	K-2
.	.	.
.	.	.
.	.	.
K-1	K	2
K	1	1

Циклическое отрицание определяется равенствами:

$[N^1_x] = [x] + 1$ при $[x] \leq K-1$

$[N^1_x] = 1$.

Симметричное отрицание по Посту определяется:

$[N^2_x] = K - [x] + 1$

Очевидно, что при K=2 циклическое и симметричное отрицания совпадают с отрицанием двузначной логики и между собой.

Операции конъюнкции и дизъюнкции определяются как минимум и максимум значений аргументов.

Темпоральные логики

Неформально, темпоральная логика — это язык, на котором можно формулировать утверждения, используя понятие времени.

Пусть есть набор переменных, которые как-то меняются со временем. Темпоральная логика позволяет формулировать утверждения типа:

- Значение a все время будет равно значению b
- Наступит момент, когда c станет нулем
- Значение d будет становится единицей бесконечно много раз

Кванторы пути:

- A — "выполнено для всех путей"
- E — "для некоторого пути"

Темпоральные операторы:

- X — "в следующий момент"
- G — "когда-нибудь, рано или поздно"
- F — "всегда, повсюду"
- U — "когда-нибудь наступит утв.2, а до него все время будет утв.1"
- R — "утв.2 будет выполнено до тех пор пока не появится утв.1"

Глава 14 Общие сведения об алгоритмах

Содержание понятия “алгоритм“ можно определить следующим образом:

➢ **Определение** *Алгоритм* - предписание, однозначно задающее процесс преобразования исходной информации в виде последовательности элементарных дискретных шагов, приводящих за их конечное число к результату.

14.1 Основные свойства алгоритма

1. *Дискретность*. Процесс решения протекает в виде последовательности отдельных действий, следующих друг за другом.

2. *Элементарность действий.* Каждое действие является настолько простым, что оно не допускает возможности неоднозначного толкования.

3. *Определенность.* Каждое действие определено и после выполнения каждого действия однозначно определяется, какое действие будет выполнено следующим.

4. *Конечность.* Алгоритм заканчивает работу после конечного числа шагов.

5. *Результативность*. В момент прекращения работы алгоритма известно, что является результатом.

6. *Массовость.* Алгоритм описывает некоторое множество процессов, применимых при различных входных данных.

Алгоритм считается правильным, если при любых допустимых данных он заканчивает работу и выдает результат, удовлетворяющий требованиям задачи. Алгоритм однозначен, если при применении к одним и тем же входным данным он дает один и тот же результат.

Схема определения алгоритма в практическом смысле выглядит так:

1. Всякий алгоритм применяется к исходным данным и выдает результирующие данные. В ходе работы алгоритма появляются промежуточные данные. Для описания данных фиксируется набор элементарных символов (алфавит данных) и

даются правила построения сложных данных из простых. Примеры простых данных: целые и действительные числа, логические переменные, символьные переменные. Примеры сложных данных: массивы, строки, структуры.

2. Данные для своего размещения требуют памяти. В ЭВМ память состоит из ячеек. Единицы объема данных и памяти согласованы, и в прикладных алгоритмических моделях объем данных можно измерять числом ячеек, в которых данные размещены.

3. Элементарные шаги алгоритма состоят из базовых действий, число которых конечно. Под ними можно подразумевать машинные команды, входящие в набор команд ЭВМ. При записи алгоритмов на языках высокого уровня в качестве базовых действий могут выступать операторы языка.

Можно выделить три крупных класса алгоритма:

вычислительные, информационные и управляющие.

Вычислительные алгоритмы, как правило, работают с простыми видами данных (числа, матрицы), но сам процесс вычисления может быть долгим и сложным.

Информационные алгоритмы представляют собой набор сравнительно небольших процедур

(например, поиск числа, слова), но работающих с большими объемами информации (базы данных). Для того чтобы они работали эффективно, важно иметь хорошую организацию данных.

Управляющие алгоритмы характерны тем, что данные к ним поступают от внешних процессов, которыми они управляют. Результатом работы этих алгоритмов являются различные управляющие воздействия.

14.2 Оценка сложности алгоритма

Сложность алгоритма помогает оценить затраты на его реализацию и определяется вычислительными мощностями, необходимыми для его выполнения. Она часто измеряется двумя параметрами: **T**(временная сложность) и **S** (пространственная сложность, или требования к памяти). И **T** и **S** обычно

представляются в виде функций от **n**, где **n** - размер входных данных.

Пусть A - алгоритм для решения некоторого класса задач, а n - размерность отдельной задачи из этого класса. В общем случае n может быть длиной обрабатываемой последовательности данных. Определим f(n) как рабочую функцию, дающую верхнюю границу для максимального числа основных операций (сложения, сравнения и т.д.), которые должен выполнить алгоритм.

Определим асимптотическую сложность алгоритма A. Если алгоритм обрабатывает входную последовательность (ВХП) размера n за время cn^2, то говорят, что временная сложность этого алгоритма - $Q(n^2)$ (читается: порядка n^2). Точный смысл этого утверждения такой: найдутся такие константы c1, c2 > 0 и такое число n0, что $c_1n^2 \le f(n) \le c_2n^2$ при всех $n \ge n_0$. Вообще, если g(n) - некоторая функция, то запись f(n) = Q(g(n)) означает, что найдутся такие c1, c2 > 0 и такое n_0, что $c_1g(n) \le f(n) \le c_2g(n)$ при всех $n \ge n_0$.

Запись f(n) = Q(g(n)) включает в себя две оценки - верхнюю и нижнюю. Их можно разделить (в данном случае нас больше интересует верхняя оценка).

Говорят, что f(n) = O(g(n)), если найдется такая константа > 0 и такое число n_0, что $0 \le f(n) \le cg(n)$ для всех $n \ge n_0$. Также существуют определения:

Функция f(n) является O[g(n)] (о-большое) для больших n, если: $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = const \neq 0$

Функция f(n) является o[g(n)] (о-малое) для больших n, если:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Говорят, что алгоритм A - *полиномиальный*, если f(n) растет не быстрее, чем полином от n. При этом в качестве g(n) оставляют член, быстрее всего растущий при увеличении n. При этом все члены низших порядков игнорируются.

Пример Если временная сложность алгоритма описывается как

$T(n) = 4n^2 + 7n + 12$, то вычислительная сложность определяется, как $O(n^2)$.

Временная сложность, определяемая таким образом не зависит от реализации. Не нужно знать ни точное время выполнения различных инструкций, ни число битов, используемых для представления различных переменных, ни даже скорость процессора. Один компьютер может быть на 50 процентов быстрее другого, у третьего шина данных может быть в два раза быстрее, но сложность алгоритма, оцененная по порядку величины, не изменится. Запись оценки сложности позволяет увидеть, как объем входных данных влияет на требования ко времени выполнения. Например, если $T(n) = O(n)$, то удвоение входных данных удвоит и время работы алгоритма. Если $T = O(2^n)$, то добавление одного бита к входным данным удвоит время выполнения.

14.3 Классификация алгоритмов по сложности

Обычно алгоритмы классифицируют в соответствии с их временной сложностью. Можно выделить следующие их типы:

1. *Постоянный* - сложность оценивается как $O(1)$.
2. *Линейный* - оценка равна $O(n)$.
3. *Квадратный* - $O(n^2)$
4. *Кубический, полиноминальный* - $O(n^3)$,$O(n^m)$.
5. *Экспоненциальный* - $O(t^{p(n)})$, t- константа, p(n) - некоторая полиномиальная функция.
6. *Факториальный* - $O(n!)$. Обладает наибольшей временной сложностью среди всех известных типов.

С ростом n временная сложность может стать настолько огромной, что это повлияет на практическую реализуемость алгоритма.

Рассмотрим таблицу, в которой сравнивается время выполнения алгоритмов разных типов при $n = 10^6$, при условии, что единицей времени для компьютера является микросекунда.

Тип	Сложность	Кол-во операций	Время при 106 операций в сек.

Постоянные	O(1)		1
	1мкс		
Линейные	O(n)	10^6	
	1 с		
Квадратичные	$O(n^2)$	10^{12}	
	11.6 дн.		
Кубические	$O(n^3)$		10^{18}
	32000 лет		
Экспоненциальные	$O(2^n)$		10^{301030}

в 10^{301006} раз больше времени существования Вселенной

14.4 Сложность проблем

Существует теория сложности, которая классифицирует не только сложность самих алгоритмов, но и сложность самих задач. Теория рассматривает минимальное время и объем памяти, необходимые для решения самого трудного варианта проблемы на теоритическом компьютере, или машине Тьюринга.

Проблемы, которые можно решить с помощью алгоритмов с полиномиальным временем, называют *решаемыми*, потому что при разумных входных данных обычно могут быть решены за разумное время (точное определение " разумности" зависит от конкретных обстоятельств).

Проблемы, которые невозможно решить за полиномиальное время, называют *нерешаемыми*, потому что нахождение их решений быстро становится невозможным. Нерешаемые проблемы иногда называют трудными.

Алан Тьюринг доказал, что некоторые проблемы принципиально неразрешимы, то есть даже отвлекаясь от временной сложности, невозможно создать алгоритм их решения.

Вот некоторые из трудных задач:

Задача комивояжера

Комивояжер должен объехать N городов с целью осуществления продажи своих товаров. Все N городов соединены дорогами по принципу "каждый с каждым". Известна стоимость проезда между двумя любыми городами. Найти оптимальный маршур движения так, чтобы побывать во всех городах и при этом иметь минимальные затраты на дорогу.

Проблема тройного брака

В комнате n мужчин, n женщин и n чиновников. Есть список разрешенных браков, записи которого состоят из одного мужчины, одной женщины и одного регистрирующего чиновника. Если дан этот список троек, то возможно ли построить n браков так, чтобы любой либо сочетался браком только с одним человеком или регистрировал только один брак?

Тройная выполнимость

Есть список n логических выражений, каждое с тремя переменными.

Например: если (x и y) то z, (x и y) или не z и т.д. Существуют ли такие значения всех переменных, чтобы все утверждения были истинными?

Пример Рассмотрим проблему вскрытия алгоритма шифрования по ключу. Временная сложность такого вскрытия пропорциональна числу возможных ключей, которое экспоненциально зависит от длины ключа. Если n - длина ключа, то сложность вскрытия грубой силой равна O(2n). При n бит сложность равна 256 и в этом случае вскрытие возможно за приемлимое время. При n = 112 бит сложность равна 2112 вскрытие становится невозможным.

Пример Рассмотрим в качестве примера задачу комивояжера. Комивояжер должен объехать N городов с целью осуществления продажи своих товаров. Все N городов соединены дорогами по принципу "каждый с каждым". Известна стоимость проезда между двумя любыми городами. Найти оптимальный маршрут движения так, чтобы побывать во всех городах и при этом иметь минимальные затраты на дорогу.

Исходная информация задана в виде перечня городов и соответствующей матрицы стоимостей, то есть двумерного массива с элементами C_{ij} , равными стоимости проезда из города i в город j. В данном случае матрица имеет N строк и N столбцов. Следует также уточнить, что маршрут начинается и заканчивается в одном (базовом) городе и не может дважды проходить через один и тот же город.

Решение (метод грубого перебора).

Произвольно пронумеруем N городов целыми числами от 1 до N, причем базовый город имеет номер N. Каждый тур (один из возможных маршрутов) однозначно соответствует перестановке целых чисел 1, 2, ..N - 1. Для каждой перестановки строим тур и определяем его стоимость. Обрабатывая все перестановки запоминаем маршрут, который имеет на текущий момент самую низкую стоимость. Если находится маршрут с меньшей стоимостью, то все дальнейшие сравнения осуществляем с ним.

Алгоритм 1. Задача комивояжера

```
TOUR ←1
MIN ← 1
for I ←1 to (N - 1)! do
P ← get P          (получение I-ой перестановки целых чисел 1,2,..N-1)
T(p),COST(T(p))    Строим тур T(P) and Вычисляем стоимость COST(T(P))
if COST(T(P)) < MIN then
TOUR←T(p),
MIN←COST(T(P))
fi
od
```

Попробуем оценить сложность данного алгоритма. Алгоритм является факториальным, с оценкой O(n!). В задаче требуется найти (N - 1)! перестановок целых чисел. Если даже требуется только один шаг для каждой перестановки, то эта часть алгоритма потребует O[(n - 1)!] шагов, поэтому любая верхняя граница для общего времени работы должна быть O(n!). Построим таблицу, иллюстрирующую вычислительную

сложность алгоритма, предполагая, что производительность компьютера 1GFLOPS (1000000000 op/s).

Кол-во городов, N	Кол-во туров	T,с	T,дн	T,лет
2	2	2e-08	<< 1	<< 1
3	6	6e-08	<<1	<< 1
4	24	2e-07	<< 1	<< 1
10	4e+06	4e-02	<< 1	<< 1
15	13e+12	1e+04	0.15	<< 1
18	6e+15	6e+07	740	20
19	1e+17	1e+09	14000	390
20	2e+18	2e+10	280000	770

Глава 15 Рекурсивные функции

Для дальнейшего рассмотрения нам понадобится ряд определений. Пусть имеются два множества X и Y.

➢ **Определение** Если некоторым элементам множества X поставлены в соответствие однозначно определенные элементы множества Y, то говорят, что задана **частичная функция** из X в Y.

Совокупность тех элементов множества X, у которых есть соответствующие элементы в Y, называется областью определения функции, а совокупность тех элементов Y, которые соответствуют элементам X, называются совокупностью значений функции.

➢ **Определение** Если область определения функции из X в Y совпадает с множеством X, то функция называется всюду определенной.

Исходная идея построения точного определения алгоритма, опирающегося на понятие рекурсивной функции, состоит в том, что любые данные (дискретные) можно закодировать натуральными числами в некоторой системе счисления, и тогда всякое их преобразование сведется к последовательности вычислительных операций, а результат обработки также будет представлять собой целое число. В данном подходе любой алгоритм, единый для данной числовой функции, вычисляет ее значение, а его элементарными шагами оказываются обычные арифметические и логические операции. Такие функции получили название вычислимых.

Пусть имеется класс функций типа $y(x_1, x_2, \ldots, x_n)$, особенностями которых является то, что все аргументы функции $x_1, \ldots, x_n$ целочисленны, и значение функции y также выражается целым числом. Другими словами, рассматриваются функции, аргументы и значения которых дискретны.

➢ **Определение** Функция $y(x_1, x_2, \ldots, x_n)$ называется эффективно вычислимой, если существует алгоритм, позволяющий вычислить ее значение по известным значениям аргументов.

Поскольку понятие алгоритма в этом определении берется в интуитивном смысле, то и понятие эффективно вычислимой

функции оказывается интуитивным. Тем не менее, при переходе от алгоритмов к вычислимым функциям возникает одно очень существенное обстоятельство.

Эта точно описанная совокупность числовых функций, совпадающая с совокупностью всех вычислимых функций при самом широком до сих пор известном понимании алгоритма, носит название совокупности *рекурсивных функций*.

Любая алгоритмическая модель и, в том числе, рекурсивные функции, должна предусматривать определение элементарных шагов алгоритма и способов построения из них какой-то последовательности преобразований, обеспечивающих необходимую последовательность обработки данных.

В рекурсивной модели такими «элементарными шагами» являются так называемые *простейшие* числовые функции S^1, 0^n и I_m^n комбинацией которых строятся все более сложные и которые *определяются* следующим образом:

$S^1(x) = x+1$ *– это одноместная* (т.е. имеет один аргумент) функция непосредственного следования;

$0^n(x_1,x_2,\ldots,x_n) = 0$ – это *n*-местная функция, тождественного равенства нулю;

$I_m^n(x_1,\ldots,x_n) = x_m$ $(1 \leq m \leq n;\ n=1,2,\ldots)$ – *n*-местная функция тождественного повторения значения одного из своих аргументов.

Перечисленные простейшие функции всюду определены и интуитивно вычислимы. Над ними определяются операции (в дальнейшем они называются *операторами*), обладающие тем свойством, что их применение к функциям, вычислимым в интуитивном смысле, порождает новые функции, также заведомо вычислимые в интуитивном смысле.

Частичные функции, которые можно получить при помощи этих операторов из простейших функций называются частично рекурсивными.

Гипотеза Черча состоит в том, что класс построенных таким образом частично рекурсивных функций совпадает с классом функций, допускающим алгоритмическое вычисление.

Переходим к рассмотрению операторов, обеспечивающих преобразование простейших функций.

15.1 Суперпозиция частичных функций

Пусть m-местные функции $f_1(x_1,\ldots, x_m)$, $f_2(x_1,\ldots, x_m),\ldots, f_n(x_1,\ldots, x_m)$ подставляются в n-местную функцию $g(x_1,\ldots, x_n)$.

В результате получается n-местная функция

$h(x_1,\ldots, x_n)=g(f_1(x_1, \ldots, x_m),\ldots, f_n(x_1,\ldots, x_m))$

➢ **Определение** Говорят, что функция h получена из функций $g, f_1, \ldots, f_n$ суперпозицией (или подстановкой).

Символически такая подстановка обозначается следующим образом: $S^{n+1}(g,f_1,\ldots,f_n)$,

где индекс сверху обозначает количество функций, подставляемых в качестве аргументов.

Если мы умеем вычислять функции $g, f_1, \ldots, f_n$, то функция h также может быть вычислена. Ясно также, что если все функции $g, f_1, \ldots, f_n$ всюду определены, то и функция h также всюду определена. Таким образом, если функции $g, f_1, \ldots, f_n$ интуитивно вычислимы, то будет интуитивно вычислимой и функция h.

Пример Найти значение $S^2(S^1,0^1)$.

Для этого значение простейшей функции 0^1 должно быть подставлено в $S^1(x)=x+1$.

Но $0^1(x)=0$, следовательно,

$h(x) = S^2(S^1, 0^1) = S^1(0^1) = 0+1= 1.$

Пример Найти значение $S^3(I_2^2,I_1^1,0^1)$.

В этом случае конечная функция будет двуместной ($n = 3–1 =2$),

следовательно

$h(x_1,x_2) = I_2^2(I_1^1,0^1) = I_2^2(x_1,0) = 0.$

15.2 Примитивная рекурсия

Пусть заданы какие-либо числовые частичные функции: n-местная $g(x_1, \ldots, x_n)$ и $(n + 2)$-местная $h(x_1,\ldots, x_n, k, y)$.

➢ **Определение** Говорят, что $(n + 1)$-местная частичная функция f образуется из функций g и h посредством примитивной рекурсии, если для всех натуральных значений $x_1,\ldots, x_n, y$ справедливо:

$$f(x_1,\ldots, x_n ,0) = g(x_1,\ldots, x_n), \qquad (1)$$

$$f(x_1,\ldots, x_n,y+1) = h(x_1,\ldots, x_n , y, f(x_1,\ldots, x_n ,0))$$

Поскольку областью определения функций является множество всех натуральных чисел, частичная функция *f*, удовлетворяющая условиям (1), существует для каждых частичных функций *g* и *h* и функция эта будет единственной.

Условия (1) задают также последовательность определения значений *f* на различных шагах рекурсии:

$$f(x_1,\ldots, x_n,0) = g(x_1,\ldots, x_n),$$
$$f(x_1,\ldots, x_n,1) = h(x_1,\ldots, x_n,1, f(x_1,\ldots, x_n,0)),$$
$$\ldots$$
$$f(x_1,\ldots, x_n,m+1) = h(x_1,\ldots, x_n, m+1, f(x_1,\ldots, x_n, m)) \quad 2)$$

Символически примитивная рекурсия обозначается ***f = R(g,h);*** в этой записи *R* рассматривается как символ двуместной частичной операции, определенной на множестве всех частичных функций.

Из соотношений (2) вытекает, в частности, что если *g* и *h* являются всюду определенными, то и *f* также является всюду определенной. Из (2) видно также то важное обстоятельство, что если мы умеем находить значения функций g и h, то значения функции $f(a_1,\ldots, a_n,m+1)$ можно вычислять «механически», находя последовательно значения на предыдущих шагах.

➢ **Определение** Частичная функция $f(x_1,\ldots, x_n)$ называется примитивно рекурсивной, если ее можно получить конечным числом операций суперпозиции и примитивной рекурсии, исходя лишь из простейших функций S^1, 0^n и I_m^n.

Если операции суперпозиции и примитивной рекурсии применить к всюду определенным функциям, в результате будет получена также всюду определенная функция. В частности, все *примитивно рекурсивные функции всюду определены.*

Пример Доказать, что двуместная функция $f(x,y)=x+y$ является примитивно-рекурсивной.

Данная функция может быть представлена в форме (1):

$$x + 0 = x = I_1^1(x)$$

$$x + (y+1) = (x+y) + 1 = S^1(x+y)$$

Следовательно, функция $f(x,y)$ образуется из примитивно рекурсивных функций операцией примитивной рекурсии и, следовательно, она сама примитивно рекурсивна.

Пример Найти значение функции $f(3,2)$, если она задана следующими соотношениями:

$$f(0,x) = 0$$
$$f(y+1,x) = f(y,x) + x$$

В данном случае $g(x) = 0$, $h(x,y,z) = y + z$.

Т.к. $f(0,x) = g(x) = 0$ при любом x, то и $f(0,2) = 0$, а другие значения можно вычислить последовательно:

$$f(1,2) = h(1,0,2) = 0 + 2 = 2$$
$$f(2,2) = h(2,2,2) = 2 + 2 = 4$$
$$f(3,2) = h(3,4,2) = 4 + 2 = 6$$

Несложно доказать, что в данном примере $f(x,y)=x \cdot y$.

15.3 Операция минимизации

Пусть задана некоторая функция $f(x,y)$. Зафиксируем значение x и выясним, при каком y значение $f(x,y)=0$. Более сложной оказывается задача отыскания *наименьшего* из тех значений y, при которых $f(x,y)=0$. Поскольку результат решения такой задачи, очевидно, зависит от x, то и наименьшее y является функцией x. Примем обозначение:

$$\varphi(x) = \mu_y[f(x,y)=0]$$

(читается: *«наименьшее y такое, что $f(x,y)=0$»*, а μ_y называет μ-оператором или оператором минимизации).

Подобным же образом определяется функция многих переменных:

$$\varphi(x_1,\ldots x_n) = \mu_y[f(x_1,\ldots x_n,y)=0]$$

Для вычисления функции φ можно предложить следующую процедуру:

Вычисляем $f(x_1,\ldots x_n,0)$; если значение равно нулю, то полагаем $\varphi(x_1,\ldots x_n)=0$.

Если $f(x_1, \ldots x_n,0) \neq 0$, то переходим к следующему шагу.

Вычисляем $f(x_1,\ldots x_n,1)$; если значение равно нулю, то полагаем $\varphi(x_1,\ldots x_n)=1$. Если $f(x_1,\ldots x_n,0) \neq 0$, то переходим к следующему шагу. И т.д.

Если окажется, что для всех y функция $f(x_1,\ldots x_n,0) \neq 0$, то функция $\varphi(x_1,\ldots x_n)$ считается неопределенной.

Пример Рассмотрим функцию $f(x,y)=x - y$, которая может быть получена с помощью оператора минимизации:

$$f(x,y) = \mu_z\ (y+z=x) = \mu_z\ [\ I_3^2(x,y,z) + I_3^3(x,y,z) = I_3^1(x,y,z)]$$

Вычислим, например, $f(7,2)$, т.е. значение функции при $y = 2$ и $x = 7$. Положим $y = 2$, а x будем придавать последовательные значения:

$z = 0$, $2 + 0 = 2 \neq 7$,

$z = 1$, $2 + 1 = 3 \neq 7$,

$z = 2$, $2 + 2 = 4 \neq 7$,

$z = 3$, $2 + 3 = 5 \neq 7$,

$z = 4$, $2 + 4 = 6 \neq 7$,

$z = 5$, $2 + 5 = 7 = 7$.

Таким образом, найдено значение функции $f(7,2) = 5$.

➢ **Определение** Частичная функция $f(x_1,\ldots, x_n)$ называется частично рекурсивной, если ее можно получить конечным числом операций суперпозиции, примитивной рекурсии и минимизации, исходя лишь из простейших функций S^1, 0^n и I_m^n.

Класс частично рекурсивных функций шире класса примитивно рекурсивных функций, т.к. все примитивно рекурсивные функции являются всюду определенными, а среди частично рекурсивных функций встречаются функции не всюду определенные, а также нигде не определенные.

Понятие частично рекурсивной функции является одним из главных понятий теории алгоритмов. Значение его состоит в следующем.

С одной стороны, каждая стандартно заданная частично рекурсивная функция вычислима путем некоторой процедуры механического характера, отвечающей интуитивному представлению об алгоритмах. С другой стороны, какие бы классы точно очерченных алгоритмов ни строились, во всех случаях неизменно оказывалось, что вычислимые посредством них числовые функции являлись частично рекурсивными. Поэтому общепринятой является научная гипотеза, формулируемая как тезис Черча:

15.4 Тезис Черча

Класс алгоритмически (или машинно) вычислимых частичных числовых функций совпадает с классом всех частично рекурсивных функций.

Этот тезис дает алгоритмическое истолкование понятие частично рекурсивной функции. Его нельзя доказать, поскольку он связывает нестрогое математическое понятие интуитивно вычислимой функции со строгим математическим понятием частично рекурсивной функции. Однако исследования, проводившиеся весьма многими математиками в течение нескольких десятилетий, выявили полную целесообразность считать *понятие частично рекурсивной функции научным эквивалентом интуитивного понятия вычислимой частичной функции.*

Тезис Черча оказался достаточным, чтобы придать необходимую точность формулировкам алгоритмических проблем и в ряде случаев сделать возможным доказательство их неразрешимости. Причина заключается в том, что обычно в алгоритмических проблемах математики речь идет не об алгоритмах, а о вычислимости некоторых специальным образом построенных функций. В силу тезиса Черча вопрос о вычислимости функции равносилен вопросу о ее рекурсивности. Понятие рекурсивной функции строгое. Поэтому обычная математическая техника позволяет иногда непосредственно доказать, что решающая задачу функция не может быть рекурсивной. Именно этим путем самому Черчу удалось доказать неразрешимость основной алгоритмической проблемы логики предикатов – проблемы тождественной истинности формул исчисления первой ступени.

Глава 16 Сложность алгоритмов

Быстрыми являются **линейные алгоритмы**, которые обладают сложностью порядка n и называются также алгоритмами порядка O(n), где n - размерность входных данных.

К линейным алгоритмам относится школьный алгоритм нахождения суммы десятичных чисел, состоящих из n_1 и n_2 цифр. Сложность этого алгоритма - $O(n_1 + n_2)$.

Есть алгоритмы, которые быстрее линейных, например, алгоритм двоичного поиска в линейном упорядоченном массиве имеет сложность $O(\log_2 n)$, n - длина массива.

Другие хорошо известные алгоритмы - деление, извлечение квадратного корня, решение систем линейных уравнений и др. - попадают в более общий класс полиномиальных алгоритмов.

➢ **Определение Полиномиальный алгоритм** (или алгоритм полиномиальной временной сложности, или алгоритм принадлежащим классу **P**) - алгоритм, у которого временная сложность равна $O(n^k)$, где k - целое число > 0.

➢ **Определение** Алгоритмы, для временной сложности которых не существует такой оценки, называются *экспоненциальными*.

➢ **Определение** Задача считается *труднорешаемой*, если для него не существует разрешающего полиномиального алгоритма.

Заметим, что при небольших значениях n экспоненциальный алгоритм может быть более быстрым, чем полиномиальный. Однако различие между этими двумя типами задач велико и всегда проявляется при больших значениях n.

16.1 Класс P

Мы называем задачу "хорошей" если для нее существует полиномиальный алгоритм. Приведем список некоторых хорошо решаемых задач.

- **Рассортировать множество из n чисел**. Сложность поведения в среднем порядка O(n log n) для быстрого алгоритма Хоара.

- **Найти эйлеровый цикл на графе из m ребер.** В силу теоремы Эйлера мы имеем необходимое и достаточное условие для существования эйлерова цикла и проверка этого условия есть алгоритм порядка O(m).
- **Задача Прима-Краскала**

Дана плоская страна и в ней n городов. Нужно соединить все города телефонной связью так, чтобы общая длина телефонных линий была минимальной.

В терминах теории графов задача Прима-Краскала выглядит следующим образом:

Дан граф с n вершинами; длины ребер заданы матрицей (d[i,j]), i,j = 1,...,n. Найти остовное дерево минимальной длины. Эта задача решается с помощью жадного алгоритма сложности O(n log n).

- **Кратчайший путь на графе, состоящем из n вершин и m ребер**. Сложность алгоритма O(m n).
- **Связные компоненты графа**. Определяются подмножества вершин в графе (связные компоненты), такие, что две вершины, принадлежащие одной и той же компоненте, всегда связаны цепочкой дуг. Если n - количество вершин, а m - количество ребер, то сложность алгоритма O(n+m).
- **Быстрое преобразование Фурье**, требующее O(n log n) арифметических операций, - один из наиболее часто используемых алгоритмов в научных вычислениях.
- **Умножение целых чисел.** Алгоритм Шёнхаге-Штрассена. Сложность алгоритма порядка O(n log n log log n). Отметим, что школьный метод для умножения двух n-разрядных чисел имеет сложность порядка $O(n^2)$.
- **Умножение матриц**. Алгоритм Штрассена имеет сложность порядка $O(n^{\log 7})$, для умножения двух матриц размера n×n. Очевидный алгоритм имеет порядок сложности $O(n^3)$.

16.2 Класс E

К экспоненциальным задачам относятся задачи, в которых требуется построить множество всех подмножеств данного

множества, все полные подграфы некоторого графа или же все поддеревья некоторого графа.

Задачи не попадающие ни в класс P, ни в класс E

На практике существуют задачи, которые заранее не могут быть отнесены ни к одному из рассмотренных выше классов. Хотя в их условиях не содержаться экспоненциальных вычислений, однако для многих из них до сих пор не разработан эффективный (т.е. полиномиальный) алгоритм.

К этому классу относятся следующие задачи:

- **задача о выполнимости**: существует ли для данной булевской формулы, находящейся в КНФ, такое распределение истинностных значений, что она имеет значение И?
- **задача коммивояжера**;
- **решение систем уравнений с целыми переменными;**
- **составление расписаний, учитывающих определенные условия**;
- **размещение** обслуживающих центров (телефон, телевидение, срочные службы) для максимального числа клиентов при минимальном числе центров;
- **оптимальная загрузка** емкости (рюкзак, поезд, корабль, самолёт) при наименьшей стоимости;
- **оптимальный раскрой** (бумага, картон, стальной прокат, отливки), оптимизация маршрутов в воздушном пространстве, инвестиций, станочного парка;
- **задача распознавания простого числа**; самый лучший в настоящее время тест на простоту имеет сложность порядка $O(L(n)^{L(L(L(n)))})$, где L(n) - количество цифр в числе n (выражение L(L(L(n))) стремиться к бесконечности очень медленно; первое число, для которого L(L(L(n))) = 2, равно $10^{999999999}$).

16.3 Недетерминированные алгоритмы

Мы собираемся более подробно классифицировать задачи, не попадающие ни в класс **P**, ни в класс **E**. Для этого вводится понятие недетерминированного алгоритма.

Неформально, мы определяем *состояние* алгоритма как комбинацию адреса выполняемой в текущий момент команды и значений всех переменных. Все алгоритмы, рассматривавшиеся до сих пор были *детерминированными*; иначе говоря, во всех них для любого данного состояния существует не больше одного вполне определенного "следующего" состояния.

Другими словами, детерминированный алгоритм в каждый момент времени может делать только что-либо одно.

В недерминированном алгоритме для любого данного состояния может быть больше одного допустимого следующего состояния; другими словами, недетерминированный алгоритм в каждый момент времени может делать больше одной вещи. Недетерминированные алгоритмы не являются в каком-то смысле вероятностными или случайными алгоритмами; они являются алгоритмами, которые могут находиться одновременно во многих состояниях.

Недетерминированный алгоритм можно моделировать с помощью *недетерминированной машины Тьюринга.*

Обобщим данное там определение, допустив, что каждое значение функции M является множеством троек {<записываемый символ>, <переход>, <номер инструкции>}. Теперь для каждого состояния машины может быть несколько следующих состояний, в соответствии с функцией перехода. И в каждом следующем состоянии запускается новая копия данной машины Тьюринга.

Очевидно, никакое физическое устройство не способно на неограниченное недетерминированное поведение; недетерминированные алгоритмы - это абстракция, которая позволяет нам игнорировать некоторые проблемы программирования поиска с возвращением.

Определим **NP** как класс всех задач, которые можно решить недетерминированными алгоритмами, работающими в течение полиномиального времени, т. е. недетерминированными алгоритмами, в которых всегда есть путь успешного вычисления за время, полиномиальное относительно входа; очевидно, $\mathbf{P} \subseteq \mathbf{NP}$. Поскольку путей вычисления может быть экспоненциально много, вероятно, что алгоритмы, допустимые в этом случае,

намного сильнее, чем детерминированные алгоритмы, допустимые для задач из **P**.

Причины, по которым задача коммивояжера попадает в класс **NP**. С оптимизационными проблемами (такими, например, как задача коммивояжера) связаны соответствующие *проблемы распознавания свойств.* Такие задачи имеют только два возможных решения - "да" или "нет". Выражаясь абстрактно, проблема распознавания T состоит просто из двух множеств: множества D_T всех возможных частных случаев (индивидуальных задач) и множества Y_T ($Y_T \subset D_T$) частных случаев с ответом "да".

Задача распознавания, соответствующая задаче о коммивояжере, может быть сформулирована следующим образом.

Условие. Заданы конечное множество $C = \{c_1, c_2, ..., c_m\}$ "городов", "расстояние" $d(c_i, c_j)$ между каждой парой городов c_i, c_j из C и граница B - положительное число.

Вопрос. Существует ли "маршрут", проходящий через все города из C, длина которого не превосходит B? Другими словами, существует ли последовательность $<c_{k(1)}, c_{k(2)}, ..., c_{k(m)}>$ элементов C такая, что

$$\sum_{i=1}^{m-1} d(c_{k(i)}, c_{k(i+1)}) + d(c_{k(m)}, c_{k(1)}) \le B?$$

Задача распознавания не может быть сложней соответствующей задачи оптимизации. Если для задачи о коммивояжере можно за полиномиальное время найти маршрут минимальной длины, то совершенно ясно как за полиномиальное время решить соответствующую задачу распознавания. Для этого только нужно найти маршрут минимальной длины, вычислить его длину и сравнить с заданной границей B.

Полиномиальный алгоритм задачи коммивояжера неизвестен.

Предположим, однако, что имеется некоторый маршрут между городами, претендующий на решение задачи распознавания. Нетрудно проверить, является ли этот маршрут

полным обходом всех городов, а если это так, то вычислить его длину, сравнить с границей B и тем самым выяснить является ли этот маршрут положительным решением задачи распознавания. Более того, эту "процедуру проверки" можно представить в виде алгоритма, временная сложность которого ограничена в виде полинома от $|I|$.

Недетерминированный алгоритм, во многих случаях, можно применить для решения задачи распознавания.

Такой алгоритм состоит из двух различных стадий - *стадии угадывания* и *стадии проверки*. По заданному частному случаю I проблемы T на первой стадии происходит просто угадывание (генерация) некоторой структуры S. Мы можем считать, что для решения задачи запускается одновременно столько копий алгоритма, сколько существует различных структур S. Затем в каждой копии I и S вместе подаются в качестве входа на стадию проверки, которая выполняется обычным детерминированным образом и либо заканчивается ответом "да", либо заканчиваются ответом "нет", либо продолжается бесконечно без остановки (два последних случая можно не различать). Недетерминированный алгоритм "решает" проблему распознавания T, если для каждого частного случая $I \in D_T$ выполнены следующие два свойства:

1. Если $I \in Y_T$, то существует такая структура S, угадывание которой для входа I приведет к тому, что стадия проверки, начиная работу на входе (I, S), закончится ответом "да".

2. Если $I \notin Y_T$, то не существует такой структуры S, угадывание которой для входа I обеспечило бы окончание стадии проверки на входе (I, S) ответом "да".

Недетерминированный алгоритм решения задачи о коммивояжере можно было бы построить, используя в качестве стадии угадывания просто выбор произвольной последовательности городов, а в качестве стадии проверки упомянутую выше "полиномиальную" процедуру" проверки маршрута. Очевидно, что для любого частного случая I найдется такая догадка S, что результатом работы стадии проверки на входе (I, S) будет "да" в том и только том случае, если для частного случая I существует маршрут искомой длины.

16.4 NP-трудные и NP-полные задачи

Различные задачи, относящие к классу NP являются эквивалентными относительно некоторого отношения, которое мы сейчас определим.

➢ **Определение.** Задача Q *полиномиально сводится* к задаче R тогда и только тогда, когда выполнены следующие условия:

- существуют функции g(x) и f(x), вычисляемые за полиномиальное время;
- для любого входа x любого частного случая задачи Q значение g(x) является входом частного случая задачи R;
- для любого решения (выхода) y задачи R значение f(y) является решением задачи Q.

Таким образом, для решения одной задачи (в данном случае - Q) используется алгоритм другой задачи (R) (рис. 12).

➢ **Определение**. Если одновременно задача Q полиномиально сводится к задаче R и R полиномиально сводится к Q, то задачи Q и R **полиномиально эквивалентны**.

➢ **Определение**. Задача является **NP-трудной (или NP-сложной**), если каждая задача из **NP** полиномиально сводится к ней.

➢ **Определение** Задача является NP-полной, если она входит в **NP** и является **NP**-трудной.

Другими словами, задача T является **NP**-трудной, если она по крайней мере так же сложна, как любая задача в **NP**.

Любая **NP**-полная задача T обладает свойством: если **P≠NP**, то T∈ **NP\P**. Точнее, T∈ **P** тогда и только тогда, когда **P=NP**.

Первой задачей, для которой было доказано, что она является **NP**-полной, проблема о выполнимости:

Условие. Дана формула исчисления высказываний F, находящаяся в конъюнктивной нормальной форме.

Вопрос. Существует ли такое распределение истинностных значений высказывательных переменных, при которых формула F выполнима?

❖ **Теорема (теорема Кука).** Задача о выполнимости является **NP**-полной.

Проблема состоит в следующем: какая-либо из этих задач имеет полиномиальную сложность? Все эти задачи эквивалентны по сложности - стоит нам найти какой-то полиномиальный алгоритм для одной из этих задач, то все эти задачи становятся полиномиально сложны.

16.5 Нормальные алгоритмы Маркова -

Алгоритм задается системой подстановок, которые указывают, какие замены символов необходимо производить и в каком порядке эти подстановки должны следовать. Такой подход был предложен А.А.Марковым. В начале 50-х годов было введено понятие *нормального алгоритма* (сам Марков называл их *алгорифмами*).

Нормальный алгоритм Маркова задается *алфавитом* А и нормальной схемой подстановок. Введем ряд определений

- **Определение Алфавит** - конечное, непустое множество элементов называемых буквами. Различные сочетания букв образуют слова.
- **Определение Слово** - это любая конечная последовательность знаков алфавита.

Марков любую последовательность букв, какую ни в одном словаре не сыщешь, называл «словами».

- **Определение** Число символов в слове называется его **длиной.**
- **Определение** Слово, длина которого равна нулю, называется ***пустым.***

Вновь рассмотрим некоторый алфавит A, содержащий конечное число знаков (букв).

- **Определение** Слово s называется ***подсловом*** слова q, если q можно представить в виде

q=rst, где **r и t** – любые слова в том же алфавите (в том числе и пустые)

В алгоритмах Маркова в качестве элементарного шага алгоритма принимается подстановка одного слова вместо другого.

Пусть в алфавите A построено исходное слово P, которое содержит подслово P_r (в общем случае таких подслов в исходном слове может быть несколько), а также имеется некоторое слово P_k в том же алфавите.

➢ **Определение Подстановкой** называется замена первого по порядку подслова P_r исходного слова P на слово P_k. Обозначается подстановка $P_r \rightarrow P_k$

Алгоритм в данной форме представления задается *системой подстановок*, которая представляет собой последовательность (список) подстановок. Если в этом списке имеются подстановки с левыми частями, которые входят в *P*, то первая из них применяется к *P*, в результате чего оно переходит в другое слово P_1. К нему вновь применяется схема подстановок и т.д. Процесс прекращается в двух случаях: либо в списке не нашлось подстановки с левой частью, входящей в P_n, либо при получении P_n была применена последняя подстановка.

➢ **Определение Нормальная схема подстановок** - это конечный набор, состоящий из пар слов, где левое слово переходит в правое (но не наоборот).

Пример Пусть задан алфавит *A ={*,1}* и единственная подстановка: **1→1*; Найти результат обработки, если исходным является слово *P = 11*111*1*

Решение

Применение нормального алгоритма с указанной подстановкой к данному слову дает последовательность (подчеркиванием выделяется преобразуемая комбинация):

11*111*1 →11111*1 →111111

т.е. алгоритм находит количество единиц в исходном слове (суммирует числа в унарной системе счисления).

Пример Алфавит содержит символы русского языка: *А ={а,б...я}*.Найти систему подстановок, обеспечивающих преобразования: *путь →муть, поло →мала*. Найти результат применения такого алгоритма к исходным словам *папа, пузо*

Решение

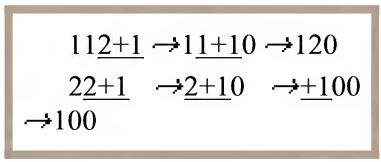
Система подстановок достаточно очевидна: *п →м, о →а.*

Применение алгоритма: папа →мапа →мама пузо →музо → муза

Пример Составить нормальный алгоритм, обеспечивающий выполнение операции сложения в троичной системе счисления

Решение

Алфавит будет содержать символы: *A ={0,1,2,+}*; система подстановок: *0+1→1, 1+1→2, 2+1→+10, +1→1*. Применим алгоритм для различных исходных слов



Различные нормальные алгоритмы отличаются друг от друга алфавитами и системами допустимых подстановок. Нормальный алгоритм Маркова можно рассматривать как стандартную форму для задания любого алгоритма. Данная форма представления алгоритма важна не только с точки зрения проведения исследований в теории алгоритмов, но она послужила основой специализированного языка символьных преобразований в системах искусственного интеллекта

Можно ли любой алгоритм представить в виде нормального алгоритма Маркова?

На этот вопрос дается ответ в виде так называемого тезиса Маркова.

❖ **Тезис Маркова**: всякий алгоритм в алфавите А представим в виде нормального алгоритма в этом же алфавите.

Это тезис потому, что его невозможно доказать, т.к. в нем фигурируют с одной стороны, интуитивное расплывчатое понятие "всякий алгоритм", а с другой стороны - точное понятие "нормальный алгоритм".

16.6 Алгоритм Евклида

Алгоритм Евклида - это способ нахождения НОД двух натуральных чисел a и b.

Предположим для определенности, что $a \geq b$. Разделим a на b с остатком:

$$a = bq + r,\ q \text{ и } r \text{ — целые, } 0 \leq r < b.$$

Теперь имеется две возможности:

1) $r = 0$. Тогда ясно, что $(a,b) = b$.

2) $r > 0$.

Тогда нужно воспользоваться следующим замечательным соотношением:

$$(a,b) = (b,r),$$

вытекающим из того, что каждый общий делитель a и b делит r и каждый общий делитель b и r делит a (это видно из определения деления с остатком), так что множество общих делителей a и b совпадает с множеством общих делителей b и r, в частности, совпадают наибольшие общие делители.

Имеем:

$$a \geq b > r.$$

Теперь вместо (a,b) надо находить (b,r). Деля b на r и обозначая остаток через r_1, мы получаем:

$$(a,b) = (b,r) = (r,r_1), \quad a \geq b > r > r_1.$$

Если $r_1 = 0$, то $(a,b) = (b,r) = r$.

Если же $r_1 \neq 0$, то надо делить r на r_1 и так далее, пока не получится остаток, равный 0.

В конце концов это произойдет, поскольку остатки все время уменьшаются.

Последний отличный от нуля остаток и будет равен (a,b).

Окончательно запишем весь процесс так:

$$a = bq + r$$

$$b = rq_1 + r_1$$

$$r = r_1q_2 + r_2$$

$$r_1 = r_2q_3 + r_3$$

$$\dots$$

$$r_{n-2} = r_{n-1}q_n + r_n$$

$$r_{n-1} = r_nq_{n+1}$$

$$a \geq b > r > r_1 > r_2 > \dots > r_{n-1} > r_n > 0$$

Тогда $(a,b) = r_n$.

Пример Вычислим $(123456789, 987654321)$.

Решение

Деля число 987654321 на число 123456789 с остатком, получим

$$987654321 = 123456789 \cdot 8 + 9.$$

Число 123456789 делится на 9.

Поэтому $(123456789, 987654321) = 9$.

Алгоритм Евклида дает гораздо больше, чем от него первоначально ожидалось получить. Из его разглядывания ясно, например, что совокупность делителей a и b совпадает с совокупностью делителей (a, b).

Еще он дает практический способ нахождения чисел u и v из $\mathbf{Z}$ (или, если угодно, из теоремы пункта 2) таких, что $r_n = au + bv = (a, b)$.

Действительно, из цепочки равенств имеем:

$r_n = r_{n-2} - r_{n-1} q_n = r_{n-2} - (r_{n-3} - r_{n-2} q_{n-1}) q_n = \ldots$

(идем по цепочке равенств снизу вверх, выражая из каждого следующего равенства остаток и подставляя его в получившееся уже к этому моменту выражение)

$\ldots = au + bv = (a, b)$.

Пример. Пусть $a = 525$, $b = 231$. Отдадим эти числа на растерзание алгоритму Евклида: (ниже приводится запись деления уголком, и каждый раз то, что было в уголке, т.е. делитель, приписывается к остатку от деления с левой стороны, а остаток, как новый делитель, берется в уголок)

```
                _    5
              525| 231
          _    462 |2
            231| 63
        _    63| 189 |3
             42 | 42
      _ 42|21  1
        42 |2
        0
```

Запись того же самого в виде цепочки равенств:

$525 = 231 \cdot 2 + 63$

$231 = 63 \cdot 3 + 42$

$63 = 42 \cdot 1 + 21$

$42 = 21 \cdot 2$

Таким образом, $(525, 231) = 21$.

Линейное представление наибольшего общего делителя:

$21 = 63 - 42 \cdot 1 = 63 - (231 - 63 \cdot 3) \cdot 1 =$

$= 525 - 231 \cdot 2 - (231 - (525 - 231 \cdot 2) \cdot 3) =$

$= 525 \cdot 4 - 231 \cdot 9,$

и наши пресловутые u и v из $\mathbf{Z}$ равны, соответственно, 4 и -9.Пункт 4 закончен.

Задачи для самостоятельного решения

1 . Предлагаю придумать два разных трехзначных числа a и b и, найти их наибольший общий делитель d и его представление в виде $d = au + bv$, $u,v \in \mathbf{Z}$.

2. Усложним себе задачу, заменив трехзначные числа четырехзначными, или даже пятизначными. Шестизначные числа брать не стоит, так как ваши родственники могут уже начать беспокоиться.

2 . К великому беспокойству родственников, все-таки найдите $d = (317811, 196418)$ и его представление в виде

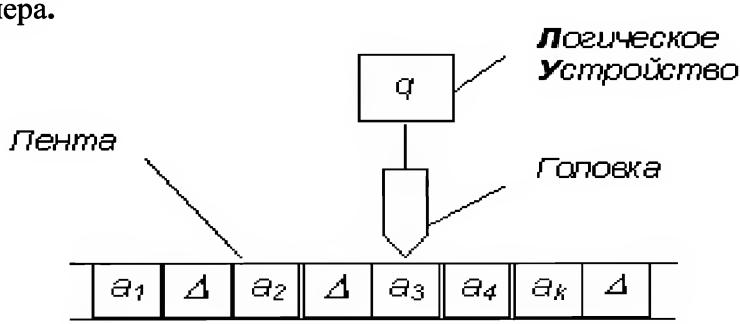
$d = 317811\, u + 196418\, v$. *

3 . Найдите $d = (81719, 52003, 33649, 30107)$.

Глава 17 *Машины Тьюринга-Поста*

Машина Тьюринга состоит из трех частей: ленты, считывающе-записывающей головки и логического устройства (см. рис. 1).

Лента выступает в качестве внешней памяти; она считается неограниченной (бесконечной) – уже это свидетельствует о том, что машина Тьюринга является модельным устройством, поскольку ни одно реальное устройство не может обладать памятью бесконечного размера.



Лента разбита на отдельные ячейки, однако, в машине Тьюринга неподвижной является головка, а лента передвигается относительно нее вправо или влево.

Другим отличием является то, что она работает не в двоичном, а некотором произвольном конечном алфавите $A = \{\Delta, a_1 ... a_n\}$ – этот алфавит называется *внешним*. В нем выделяется специальный символ – , называемый *пустым знаком* – его посылка в какую-либо ячейку стирает тот знак, который до этого там находился, и оставляет ячейку пустой.

В каждую ячейку ленты может быть записан лишь один символ.

Информация, хранящаяся на ленте, изображается *конечной* последовательностью знаков внешнего алфавита, отличных от пустого знака.

Головка всегда расположена над одной из ячеек ленты. Работа происходит тактами (шагами). Система исполняемых головкой команд предельно проста: на каждом такте она производит замену знака в обозреваемой ячейке a_i знаком a_j. При этом возможны сочетания:

- $j = i$ – это означает, что в обозреваемой ячейке знак не изменился;
- $i \neq 0$, $j = 0$ означает, что хранившийся в ячейке знак заменяется пустым, т.е. стирается;
- $i = 0$, $j \neq 0$ означает, что пустой знак заменяется непустым (с номером j в алфавите), т.е. производится вставка знака;
- $i \neq j \neq 0$ соответствует замене одного знака другим.

Таким образом, в машине Тьюринга реализуется система предельно простых команд обработки информации.

Эта система команд обработки дополняется также предельно простой системой команд перемещений ленты: на ячейку влево, на ячейку вправо и остаться на месте, т.е. адрес обозреваемой ячейки в результате выполнения команды может либо измениться на 1, либо остаться неизменным. Однако, хотя фактически происходит перемещение ленты, обычно рассматривается *сдвиг головки* относительно обозреваемой секции – по этой причине команда сдвига ленты влево обозначается

R (*«Right»*), сдвига вправо

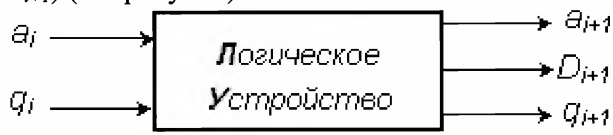
– L (*«Left»*),

отсутствие сдвига – S (*«Stop»*).

В дальнейшем мы будем говорить именно о сдвиге головки и считать R, L и S командами ее движения. Элементарность этих команд означает, что при необходимости обращения к содержимому некоторой ячейки, она отыскивается только посредством цепочки отдельных сдвигов на одну ячейку. Разумеется, это значительно удлиняет процесс обработки, зато позволяет обойтись без нумерации ячеек и использования команд перехода по адресу, т.е. сокращает количество истинно элементарных шагов, что важно в теоретическом отношении.

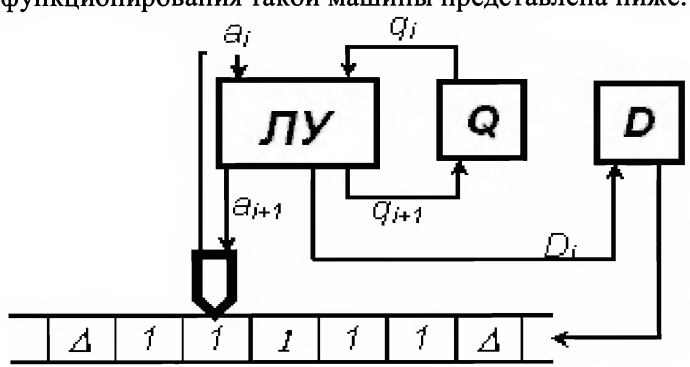
Обработка информации и выдача команд на запись знака, а также сдвига ленты в машине Тьюринга производится

логическим устройством (ЛУ). ЛУ может находиться в одном из состояний, которые образуют конечное множество и обозначаются $Q = \{q_1 \ldots q_m, z\}$, причем, состояние z соответствует завершению работы, а q_1 является начальным (исходным). Q совместно со знаками R, L, S образуют *внутренний алфавит* машины. *ЛУ* имеет два входных канала (a_i, q_i) и три выходных (a_{i+1}, q_{i+1}, D_{i+1}) (см. рисунок):



Понимать схему необходимо следующим образом: на такте i на один вход *ЛУ* подается знак из обозреваемой в данный момент ячейки (a_i), а на другой вход – знак, обозначающий состояние *ЛУ* в данный момент (q_i). В зависимости от полученного сочетания знаков (a_i, q_i) и имеющихся правил обработки *ЛУ* вырабатывает и по первому выходному каналу направляет в обозреваемую ячейку новый знак (a_{i+1}), подает команду перемещения головки (D_{i+1} из R, L и S), а также дает команду на вызов следующего управляющего знака ($qi+1$).

Таким образом, элементарный шаг (такт) работы машины Тьюринга заключается в следующем: головка считывает символ из обозреваемой ячейки и, в зависимости от своего состояния и прочитанного символа, выполняет команду, в которой указано, какой символ записать (или стереть) и какое движение совершить. При этом и головка переходит в новое состояние. Схема функционирования такой машины представлена ниже.



В данной схеме отражено разделение памяти на *внешнюю* и *внутреннюю*. Внешняя представлена, как указывалось, в виде бесконечной ленты – она предназначена для хранения информации, закодированной в символах внешнего алфавита.

Внутренняя память представлена двумя ячейками для хранения следующей команды в течение текущего такта: в Q передается из *ЛУ* и сохраняется следующее состояние (q_{i+1}), а в D – команда сдвига (D_{i+1}). Из Q по линии обратной связи q_{i+1} поступает в *ЛУ*, а из D команда поступает на исполнительный механизм, осуществляющий при необходимости перемещение ленты на одну позицию вправо или влево.

Общее правило, по которому работает машина Тьюринга, можно представить следующей записью: $q_i a_j \rightarrow q_i' a_j' D_k$, т.е. после обзора символа a_j головкой в состоянии q_i, в ячейку записывается символ a_j', головка переходит в состояние q_i', а лента совершает движение D_k. Для каждой комбинации $q_i a_j$ имеется *ровно одно* правило преобразования (правил нет только для z, поскольку, попав в это состояние, машина останавливается).

Это означает, что логический блок реализует функцию, сопоставляющую каждой паре входных сигналов $q_i a_j$ одну и только одну тройку выходных $q_i' a_j' D_k$ – она называется *логической функцией машины* и обычно представляется в виде таблицы (*функциональной схемой машины*), столбцы которой обозначаются символами состояний, а строки – знаками внешнего алфавита.

Если знаков внешнего алфавита n, а число состояний *ЛУ* m, то, очевидно, общее число правил преобразования составит $n \cdot m$.

Конкретная машина Тьюринга задается перечислением элементов множеств A и Q, а также, логической функцией, которую реализует *ЛУ*, т.е. набором правил преобразования. Ясно, что различных множеств A, Q и логических функций может быть бесконечно много, т.е. и машин Тьюринга также бесконечно много.

Прежде, чем обсуждать функционирование машины Тьюринга, введем еще одно понятие.

- **Определение Конфигурация машины -** совокупность состояний всех ячеек ленты, состояния ЛУ и положение головки .

Записать конфигурацию можно следующим образом: $\Delta a_1 q_i a_j ... a_k \Delta$, которая означает, что в слове из k символов обозревается секция номер j и при этом управляющее устройство находится в состоянии q_i. Ясно, что конфигурация машины может содержать любое количество символов внешнего алфавита и лишь один символ внутреннего. Часто конфигурацию записывают в виде $\alpha_1 q_i \alpha_2$, где α_1 – слово на ленте слева от головки, α_2 – слово на ленте справа от головки, включая обозреваемый знак. Слева от α_1 и справа от α_2 лента пуста. Конфигурация может быть записана следующим образом: $a_1 \Delta a_2 \Delta q a_3 a_4 a_k$, $1q1111$.

Перед началом работы на пустую ленту записывается исходное слово α конечной длины в алфавите A; головка устанавливается над первым символом слова α, *ЛУ* переводится в состояние q_1 (т.е. начальная конфигурация имеет вид $q_1\alpha$). Поскольку в каждой конфигурации реализуется только одно правило преобразования, начальная конфигурация однозначно определяет всю последующую работу машины, т.е. всю последовательность конфигураций вплоть до прекращения работы.

В зависимости от начальной конфигурации возможны два варианта развития событий:

- после конечного числа тактов машина останавливается по команде остановки; при этом на ленте оказывается конечная конфигурация, соответствующая выходной информации;
- остановки не происходит.

В первом случае говорят, что данная машина применима к начальной информации, во втором – нет. Вся совокупность входных конфигураций, при которых машина обеспечивает получение результата, образуют *класс решаемых задач*. Очевидно, применять машину Тьюринга для задачи, не входящей в класс решаемых, бессмысленно.

С другой стороны, во многих случаях возможно расширение класса решаемых задач за счет создания другой машины

Тьюринга. Возникает вопрос: можно ли построить такую универсальную машину (хотя бы на теоретическом уровне), которая решала бы любую задачу? Здесь мы подошли к вопросу об алгоритмической разрешимости, который будет исследован позднее.

Пример Рассмотрим решение обсуждавшейся в предыдущем параграфе задачи о добавлении *1* к унарному числу посредством машины Тьюринга. Внешний алфавит может быть задан множеством $A=\{\Delta,1\}$, где *1* соответствует заполненной секции, а Δ – пустому знаку, причем заполненные следуют друг за другом подряд. Внутренний алфавит задается множеством $Q=\{q,z\}$, где q соответствует рабочему состоянию *ЛУ*, а z – остановке. Набор всех правил преобразования (логическая функция) может быть представлен *функциональной схемой:*

A	q	z
Δ	z1S	zΔS
1	q1R	z1S

Составляется функциональная схема в виде таблицы таким образом, что знаки, обозначающие колонки и строки, определяют входные параметры *ЛУ*, а в ячейке таблицы на их пересечении стоит выходная команда. В частности, если головка машины обозревает секцию ленты со знаком *1* и машина находится в рабочем состоянии (q), то результатом ее работы на данном такте должно стать повторение *1* в данной секции и переход на одну секцию вправо R (при этом, как указывалось, лента сдвигается влево) – эта команда записывается как *q1R*. Если же в обозреваемой секции Δ, а состояние *ЛУ* q, то Δ будет заменен *1*, сдвига ленты производиться не будет и машина остановится – *z1S*. При комбинации на входе Δz, как и *1z*, машина находится в нерабочем состоянии – не происходит ни изменения конфигурации, ни движения – по этой причине такие комбинации в функциональных схемах в дальнейшем отображаться не будут.

Пусть начальной является конфигурация *1q1111*. Незначащие пустые секции слева и справа от заполненных в конфигурацию не включаются; поскольку в данной задаче несколько заполненных секций следуют подряд, только они и указываются в конфигурации

Тогда работа машины в соответствии с описанной логической функции будет происходить следующим образом:

Такт 1 Обозревается *1*, в *ЛУ* состояние *q*. Выходная команда *q1R*, что эквивалентно перемещению головки по отношению ленты на 1 шаг вправо. Следовательно, образуется промежуточная конфигурация *11q111*.

Такт 2 – аналогичным образом осуществляется переход к конфигурации *111q11*

Такт 3 – переход к конфигурации *1111q1*

Такт 4 –переход к конфигурации *11111q* Δ(здесь для лучшего понимания правый Δуказан в явном виде).

Такт 5 Обозревается Δ, в *ЛУ* состояние *q*. Выходная команда *z1S* – вместо Δв ячейку записывается *1*, сдвига нет, работа прекращается. Конечная конфигурация *111111z*

Задача решена

Пример Имеется запись многоразрядного целого числа *n* в десятичной системе счисления; построить машину Тьюринга, которая обеспечивала бы вычисление значение *n+1*

Используем внешний алфавит $A=\{0,1,...,9,\Delta\}$, в котором символ Δсоответствует пустому знаку. Внутренний алфавит, как и в предыдущей задаче, образуется двумя состояниями – рабочим (*q*) и остановкой (*z*) ($Q=\{q, z\}$). Исходное число *n*, а также результат – *n+1* – записываются в десятичной системе, причем, цифры размещаются по одной в соседних ячейках без пропусков.

Функциональную схему представляется таблицей (для удобства строка будут соответствовать состоянию *q*, а столбцы – знакам внешнего алфавита)

0	1	2	3	4	5	6	7	8	9	Δ
z1 S	z2 S	z3 S	z4 S	z5 S	z6 S	z7S	z8S	z9S	q0 L	z1 S

Пусть начальной конфигурацией будет *21q9*

Такт 1 *q9→q0L*, т.е. *9* будет заменена на *0* и головка сдвинется на разряд десятков – промежуточная конфигурация *2q10*

Такт 2 *q1→z2S*, т.е. *1* будет заменена на *2* и произойдет остановка с конечной конфигурацией *2z20*, т.е. получен результат сложения *219+1*

Пусть начальной будет конфигурация *99q9*

Такт 1 *q9→q0L*, т.е. сформируется промежуточная конфигурация *9q90*

Такт 2 *q9→ q0L* – возникнет конфигурация *q900*

Такт 3 *q9→ q0L* – возникнет *q∆000*

Такт 4 *q∆ →z1S* – возникнет *z1000* и работа прекращается

Таким образом, описанный алгоритм действительно обеспечивает суммирование любого целого десятичного числа и единицы. Ясно также, что при необходимости произвести сложение не с единицей, а с каким-то целым *m*, то данный алгоритм необходимо повторить *m* раз. Умножение целых чисел также может быть сведено к сложению числа с самим собой. Следовательно, машины Тьюринга обладают важным свойством – возможностью построения новой машины путем объединения уже имеющихся – такая операция называется *композицией*.

По своему устройству машина Тьюринга крайне примитивна. Она намного проще самых первых компьютеров. Примитивизм состоит в том, что у нее предельно прост набор элементарных операций, производимых головкой – считывание и запись, а также в том, что доступ к ячейкам памяти (секциям ленты) в ней происходит не по адресу, как в компьютерах, а путем последовательного перемещения вдоль ленты. По этой причине даже такие простые действия как сложение или сравнение двух символов машина Тьюринга производит за несколько шагов, а обычные операции сложения и умножения требуют весьма большого числа элементарных действий. Однако машина Тьюринга была придумана не как модель (прототип) реальных вычислительных машин, а для того, чтобы показать принципиальную (теоретическую) возможность построения сколь угодно сложного алгоритма из предельно

простых операций, причем сами операции и переход от одной к последующей машина выполняет автоматически.

17.1 Тезис Тьюринга

Машина Тьюринга дает один из путей уточнения понятия алгоритма. В связи с этим возникают вопросы:

- насколько общим является понятие машины Тьюринга?
- можно ли считать, что способ задания алгоритмов с помощью машины Тьюринга является универсальным?
- может ли всякий алгоритм задаваться таким образом?

На эти вопросы современная теория алгоритмов предлагает ответ в виде следующей гипотезы:

Всякий алгоритм может быть задан посредством тьюринговой функциональной схемы и реализован в соответствующей машине Тьюринга

Эта гипотеза получила название *тезиса Тьюринга*. Как и тезис Черча, ее нельзя доказать, так как она связывает нестрогое определение понятия алгоритма со строгим определением машины Тьюринга.

В принципе, эта гипотеза может быть опровергнута, если удастся привести пример алгоритма, который не может быть реализован с помощью тьюринговой функциональной схемы.

Однако все известные до сих пор алгоритмы могут быть заданы посредством тьюринговых функциональных схем.

17.2 Возможности машин Тьюринга

Богатство возможностей конструкции Тьюринга проявляется в том, что если какие-то алгоритмы *A* и *B* реализуются машинами Тьюринга, то можно строить программы машин Тьюринга, реализующие различные композиции алгоритмов *A* и *B*, например, «выполнить *A*, затем выполнить *B*» или «Выполнить *A*. Если в результате получилось слово *да*, то выполнить *B*. В противном случае не выполнять *B*» или «Выполнять поочередно *A*, *B*, пока *B* не даст ответ 0».

В интуитивном смысле такие композиции являются

алгоритмами. Поэтому их реализация посредством машины Тьюринга служит одним из способов обоснования универсальности конструкции Тьюринга.

Реализуемость таких композиций доказывается в общем виде, независимо от особенностей конкретных алгоритмов A и B. Доказательство состоит в том, что указывается способ построения из программ A и B программы нужной композиции.

Пример, Построим машину A. B, эквивалентную последовательному выполнению алгоритмов A и B.

Машина A имеет m состояний $q_1, \ldots, q_m$;

машина B имеет k состояний $q_1, \ldots, q_m$.

Переименовываем состояния машины B, заменяя q_1 на q_{m+1}, q_2 на q_{m+2}, ... , q_k на q_{m+k}.

Соответственно заменяем и все ссылки на состояния в клетках программы B. В программе A всюду знак ! заменяем на указание состояния q_{m+1}. Записываем полученную программу A, а под ней программу B с переименованными состояниями. Вместе они образуют искомую программу $A \bullet B$. Пока выполняется алгоритм A, в программе $A \bullet B$ работает часть A без учета части B. Когда алгоритм A дойдет до конца, то вместо останова произойдет переход в первое состояние части B, и затем часть B будет работать обычным образом, как будто части A и не было.

Пример Построить сложную машину Тьюринга для подсчета на ленте штрихов, которые располагаются подряд и образуют входное слово. Нужно стереть все штрихи и написать на ленту их число, представленное в десятичной системе.

Будем формировать это число на ленте слева от штрихов. В начальный момент машина Тьюринга обозревает любой из штрихов и находится в состоянии q_1.

Для рассматриваемой задачи схема программы может выглядеть так:

1° . Найти правый конец слова на ленте.

2°. Если слово оканчивается штрихом, то стереть этот штрих, иначе остановить машину.

3°. Прибавить к числу единицу и перейти к п. 1°.

Каждый раз стирается самый правый штрих и к числу

прибавляется единица.

Выполнение этих трех пунктов повторяется до тех пор, пока не будет стерт последний штрих, после чего, согласно условию п. 2°, . машина Тьюринга остановится.

Каждый из этих пунктов может быть реализован одним состоянием машины Тьюринга. Итак, нам понадобятся три состояния машины Тьюринга. В состоянии q_1 автомат будет искать правый конец слова; q_2 будет состоянием стирания штрихов; q_3 будет состоянием прибавления к числу единицы.

В таблице приводится программа предлагаемой машины Тьюринга.

	Λ	0	1	2		8	9	/
q_1	Л, q_2	П, q_1	П, q_1	П, q_1		П, q_1	П, q_1	П, q_1
q_2	!	!	!	!		!	!	Λ,Л,q_3
q_3	1,П,q_1	1,П,q_1	2,П,q_1	3,П,q_1		9,П,q_1	0,Л,q_3	/,Л,q_3

Машина «видит» на ленте цифры, которые она писала сама, и штрихи, находящиеся там с самого начала. В состоянии q_1 признаком достижения правого конца слова служит пустая обозреваемая ячейка; при этом автомат сдвигается по ленте на шаг влево (т.е. начинает обозревать самый правый непустой символ) и переходит в состояние q_2. Находясь в состоянии q_2 и увидев штрих, автомат стирает его, сдвигается на шаг влево и переходит в состояние q_3 прибавления единицы.

Если же автомат в состоянии q_2 видит цифру, то машина останавливается, так как это означает, что все штрихи уже стерты. В состоянии q_3 автомат двигается по ленте влево, минуя оставшиеся штрихи, пока не дойдет до числа, и прибавляет к числу единицу аналогично тому, как это делалось в предыдущем примере.

В начальный момент автомат в состоянии q_1 может находиться против любого штриха из входного слова.

Например, пусть входное слово состоит из трех штрихов и автомат первоначально находится против среднего штриха:

Λ	/	/	/	Λ
		!		
		q_1		

Начав работать, автомат сдвинется два раза вправо в состоянии q_1 после чего возникнет ситуация:

Λ	/	/	/	Λ
				!
				q_1

в этот момент автомат сдвигается влево и переходит в состояние q_2:

Λ	/	/	/	Λ
			!	
			q_2	

Затем обозреваемый штрих стирается, автомат сдвигается влево и переходит в состояние q_3:

Λ	/	/	Λ	Λ
		!		
		q_3		

стирается, автомат сдвигается влево

Затем он движется влево, оставаясь в состоянии q_3, пока не увидит пустую ячейку, после чего записывает туда цифру 1, сдвигается вправо и переходит в состояние q_1:

1	/	/	Λ	Λ
	!			
	q_1			

Далее в состоянии q_1 автомат движется вправо до первой пустой ячейки, увидев которую, сдвигается влево и переходит в состояние q_2:

1	/	/	Λ	Λ

		!		
		q_2		

Очередной штрих стирается, автомат сдвигается влево и переходит в состояние q_3:

1	/	Λ	Λ	Λ
	!			
	q_3			

Еще один сдвиг влево в состоянии q_3, и автомат заменяет цифру 1 на 2, сдвигается вправо и переходит в состояние q_1:

2	/	Λ	Λ	Λ
	!			
	q_1			

Снова сдвиг вправо, и следующая смена состояния со сдвигом влево:

2	/	Λ	Λ	Λ
	!			
	q_2			

Стирание штриха (последнего), сдвиг влево и переход в состояние q_3:

2	Λ	Λ	Λ	Λ
!				
q_3				

После этого цифра 2 заменяется на 3, автомат сдвигается вправо, переходя в состояние q_3:

3	Λ	Λ	Λ	Λ
	!			
	q_1			

Далее следует сдвиг влево с переходом в состояние q_2, машина останавливается, оставив на ленте выходное слово.

Пример *A* - алгоритм подсчета штрихов на ленте, а *B* - алгоритм прибавления единицы к числу на ленте, то мы можем

воспользоваться уже рассмотренными программами машин Тьюринга, реализующими эти алгоритмы. В данном случае $m=3$ и $k=1$. Из программы A получаем первые три строки программы $A \bullet B$.

Последняя, четвертая строка программы $A \bullet B$ получается из программы B.

	Λ	0	1	2	…	8	8	/
q_1	Л, q_2	П, q_1	П, q_1	П, q_1		П, q_1	П, q_1	П, q_1
q_2	q_4	q_4	q_4	q_4		q_4	q_4	Λ,Л,q_3
q_3	1,П,q_1	1,П,q_1	2,П,q_1	3,П,q_1		9,П,q_1	0,П,q_1	/,П,q_3
q_4	1, !	1, !	2, !	3, !		9, !	0,Л,q_4	

Полученная программа $A \bullet B$ описывает машину Тьюринга, которая сначала подсчитывает число штрихов на ленте, стирая их, a затем прибавляет к этому числу единицу. Заметим, что в программе $A \bullet B$ осталась незаполненной клетка на пересечении строки q_4 и столбца /, но эта клетка никогда не будет использоваться, так как алгоритм B не имеет дела со знаком /.

Аналогично конструируются и другие композиции машин Тьюринга; каждый раз строятся общие правила: что на что менять в исходных программах.

Описывая различные алгоритмы для машин Тьюринга и доказывая реализуемость всевозможных композиций алгоритмов, Тьюринг убедительно показал разнообразие возможностей предложенной им конструкции, что позволило ему выступить со следующим тезисом:

Всякий алгоритм может быть реализован соответствующей машиной Тьюринга.

Это *основная гипотеза теории алгоритмов* в форме Тьюринга (здесь не оговаривается «всякий алгоритм преобразования слов», 'Так как мы уже ранее договорились, что действие любого алгоритма сводится к преобразованию слов).

Одновременно этот тезис является формальным определением алгоритма. Теперь можно доказывать существование или несуществование алгоритмов, описывая соответствующие машины Тьюринга или доказывая невозможность их построения. Этим для нас открывается общий подход к поиску алгоритмических решений.

Если поиск решения наталкивается на препятствие, то мы пытаемся использовать это препятствие для доказательства невозможности решения, опираясь на основную гипотезу теории алгоритмов. Если же при доказательстве невозможности возникает свое препятствие, то оно может помочь нам продвинуться в поиске решения, хотя бы частично устранив прежнее препятствие. Так, поочередно пытаясь доказать то существование, то несуществование решения, мы можем постепенно приблизиться к пониманию существа стоящей перед нами задачи.

Доказать тезис Тьюринга нельзя, так как в его формулировке не определено понятие «всякий алгоритм», т. е. левая часть тождества. Его можно только обосновать, представляя различные известные алгоритмы в виде машин Тьюринга.

Дополнительное обоснование этого тезиса состоит в том, что позднее было предложено еще несколько общих определений понятия алгоритма и каждый раз удавалось доказать, что, хотя новые алгоритмические схемы и выглядят иначе, они в действительности эквивалентны машинам Тьюринга: все, что реализуемо в одной из этих конструкций, можно сделать и в других. Эти утверждения доказываются строго, так как в них речь идет уже о тождественности формальных схем. Даже когда предпринимались специальные попытки выйти за рамки машин Тьюринга, строя алгоритмические схемы, казалось бы, противоречащие понятию машины Тьюринга, в конечном итоге все же оказывалось, что эти схемы сводятся к машинам Тьюринга.

До сих пор мы имели дело со специализированными машинами Тьюринга, предназначенными для решения конкретных задач. Однако, рассмотренный нами общий способ интерпретации работы машин Тьюринга сам является алгоритмом, а стало быть и ему должна соответствовать некоторая машина Тьюринга, в которой входное слово состоит из изображения программы и входного слова интерпретируемой машины. Такая машина называется *универсальной,* так как она способна выполнять задания для любой машины Тьюринга. Мы не будем описывать здесь программу универсальной машины

Тьюринга, а рассмотрим только, как можно представлять информацию на ее ленте.

Чтобы получить изображение программы интерпретируемой машины, нужно закодировать эту программу в алфавите универсальной машины. Программа кодируется и записывается на ленту универсальной машины последовательно, строка за строкой.

Алфавит универсальной машины Тьюринга содержит буквы, используемые при записи программ, включая и алфавит интерпретируемой машины.

Кроме того, в него входят знаки препинания запятая (,) двоеточие (:) и точка(.) для разделения частей линейного изображения программы; знак * отделяет изображение программы от входного слова для интерпретируемой машины, а знак ↑ служит для указания положения автомата интерпретируемой машины.

Например, рассмотрим, как интерпретируется на универсальной машине Тьюринга применение к числу 199 рассмотренной нами машины для прибавления единицы. Входное слово на ленте универсальной машины будет выглядеть так:

Λ', 0, 1, , 8, 9, q_1 . 1, ! . 1, ! . 2, !. . 9, ! . 0, Л, q_1 * *199*↑

Здесь левее знака * изображена программа машины Тьюринга для прибавления единицы, причем сначала перечислены буквы алфавита этой машины. (Мы заменили Λ на Λ', чтобы этот знак пустой ячейки на ленте интерпретируемой машины не путать с обозначением пустой ячейки на ленте универсальной машины.)

Предполагается, что автомат универсальной машины предварительно установлен против самой левой буквы ее входного слова(в нашем случае против знака Λ'). Мы не указываем, с какого состояния начинается работа интерпретируемой машины, так как ранее договорились, что это всегда состояние q_1.

На самом деле нет необходимости в том, чтобы алфавит универсальной машины Тьюринга включал все знаки алфавита интерпретируемой машины; вместо этого знаки

интерпретируемого алфавита можно кодировать небольшим числом знаков алфавита универсальной машины. (Для такой кодировки в принципе достаточно двух знаков).

После завершения работы универсальной машины на ее ленте должно остаться то слово, которое получилось бы в результате работы интерпретируемой машины; в нашем случае это слово 200

Если интерпретируемая машина не применима к какому-то слову, то универсальная машина тоже должна быть неприменимой, т. е. должна работать над кодировкой этого слова бесконечно долго.

17.3 Алгоритмическая машина Поста

На самом деле, Пост, в отличие от Тьюринга, не пользовался термином *«машина»*, а называл свою модель *алгоритмической системой*. Мы же, как принято в литературе, все же будем говорить о *машине Поста*, подчеркивая тем самым единство подходов обоих авторов.

Абстрактная машина Поста состоит из бесконечной ленты, разделенной на равные секции, а также считывающе-записывающей головки. Каждая секция может быть либо пуста (т.е. в нее ничего не записано), либо заполнена (*отмечена* – т.е. в нее записана *метка*).

Вводится понятие *состояние ленты* как информация о том, какие секции пусты, а какие отмечены (по-другому: состояние ленты – это распределение меток по секциям, т.е. это функция, которая каждому числовому номеру секции ставит в соответствие либо метку, либо знак *«пусто»*). Естественно, в процессе работы машины состояние ленты меняется. Состояние ленты и информация о положении головки характеризуют *состояние машины* Поста.

Условимся обозначать головку знаком *«▽»* над обозреваемой секцией, а метку – знаком *«М»* внутри секции. Пустая секция никакого знака не содержит.

За один такт (его называют *шагом*) головка может сдвинуться на одну секцию вправо или влево и поставить или удалить метку.

Работа машины Поста заключает в переходе от одного состояния машины к другому в соответствии с заданной программой, которая строится из отдельных команд. Каждая команда имеет следующую структуру:

xKy, где *x* – номер исполняемой команды; *K* – указание о выполняемом действии; *y* – номер следующей команды (*наследника*).

Система команд машины включающая шесть действий, представлена в таблице. Данный перечень должен быть дополнен следующими условиями:

команда <*xMy*> может быть выполнена только в пустой секции;

команда <*xCy*> может применяться только к заполненной секции;

номер наследника любой команды *(y)* должен соответствовать номеру команды, обязательной имеющейся в данной программе.

Таблица

№ п/п	Команда	Запись команды	Описание действий машины
1	Шаг вправо	$X \rightarrow y$	Сдвиг головки на одну секцию вправо
2	Шаг влево	$X \leftarrow y$	Сдвиг головки на одну секцию влево
3	Установить метку	*XMy*	В обозреваемую секцию ставится метка
4	Стереть метку	*XCy*	Из обозреваемой секции удаляется метка
5	Передача управления	$x \begin{matrix} \nearrow y_1 \\ \searrow y_2 \end{matrix}$	При отсутствии метки в обозреваемой секции управление передается команде y_1, при наличии – команде y_2.
6	Остановка	*x стоп*	Прекращение работы машины

Если данные условия не выполняются, происходит *безрезультатная* остановка машины, т.е. остановка до получения запланированного результата.

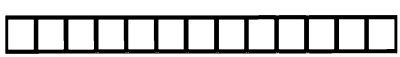
В отличие от этой ситуации, остановка по команде *<x стоп>* является *результативной*, т.е. она происходит после того, как результат действия алгоритма получен.

Кроме того, возможна ситуация, когда машина не останавливается никогда – это происходит, если ни одна из команд не содержит в качестве последователя номера команды остановки или программа не переходит к этой команде.

Еще одним исходным соображением является следующее: поскольку знаки любого конечного алфавита могут быть закодированы цифрами, преобразование исходного слова может быть представлено в виде некоторых правил обработки чисел. По этой причине в машине Поста предусматривается только запись (представление) целых положительных чисел.

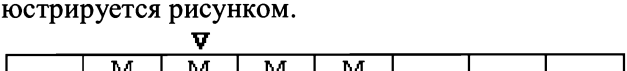
Целое число k записывается на ленте машины Поста посредством $k+1$ следующих подряд отмеченных секций, т.е. применяется *унарная* система счисления.

Соседние записи чисел на ленте разделяются одной или несколькими пустыми секциями. Ниже приведен пример записи чисел 0, 2 и 3.

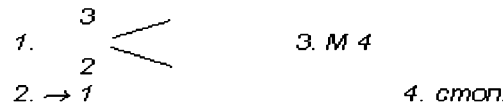


Круг вычислительных задач, решаемых с помощью машины Поста, весьма широк. Однако как указывалось выше, на уровне элементарных шагов все сводится к постановке или удалению метки и сдвигу головки. В качестве примеров рассмотрим несколько задач, традиционно обсуждаемых при освоении машины Поста. Поскольку вид программы (последовательности команд машины) зависит от начального состояния машины, оно должно быть в явном виде указано в постановке задачи.

Пример На ленте записано некоторое число, и головка обозревает одну из помеченных секций (любую). Составить программу прибавления единицы к этому числу. Ситуация иллюстрируется рисунком.



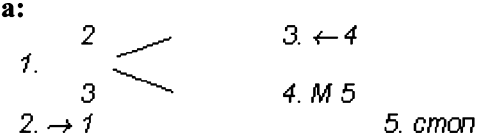
Программа, обеспечивающая решение задачи, состоит из 4-х команд:



Последовательное исполнение команд 1 и 2 приводит к тому, что головка за два такта работы машины сдвигается на одну позицию вправо. Это передвижение продолжается до тех пор, пока после очередного сдвига под головкой не окажется пустой ячейки – тогда по команде 3 в нее будет поставлена метка и по команде 4 машина остановится.

Пример На ленте записано некоторое число, и головка обозревает одну из свободных секций (любую) левее записи. Составить программу прибавления единицы к этому числу.

Программа:



Комментарий к работе программы подобен приведенному выше с той лишь разницей, что метка ставится перед исходным числом.

Можно показать, что с помощью машины Поста реализуются (хотя и довольно громоздко) все арифметические действия над числами в унарной системе счисления.

Машина Поста обеспечивает весьма неплохую и полезную практику программирования. Недостатком оказывается чисто теоретический (т.е. непроверяемый) характер программ, однако он достаточно легко преодолевается, если построить эмуляцию машины на каком-либо языке программирования

Числа же, как было показано ранее, могут быть использованы для кодирования любой дискретной информации.

В частности, состояние ленты можно представить словом в двоичном алфавите, где *0* будет соответствовать пустой секции, а *1* – отмеченной. В процессе работы меняется состояние ленты и, следовательно, от исходного слова происходит переход к выходному, представленному в том же двоичном алфавите.

Глоссарий

Вычислимые функции - множество функций вида $f: N \to N$, которые могут быть реализованы на исполнителе машин Тьюринга.

Дедукция – метод вывода правильных заключений из посылок.

Классическая логика – логика, основанная на силлогизмах.

Математическая логика - современная форма логики, которая полностью опирается на формальные математические методы. Она изучает только умозаключения со строго определенными объектами и суждениями, для которых можно однозначно решить, истинны они или ложны.

Модус - разновидность некоторой общей схемы рассуждения, разновидность силлогизмов.

Парадоксы - ситуация или высказывание, утверждение, суждение или вывод, которые могут существовать в реальности, но не иметь логического объяснения.

Силлогизм - рассуждения, в котором из заданных двух суждений выводится третье.

Силлогистика – теория правильных рассуждений.

Символическая логика — направление в математической логике, изучающее формальные системы посредством построения формализованных языков.

Формальная логика - направление в математической логике связанное с анализом обычных содержательных умозаключений, выражаемых разговорным языком.

Алфавит исчисления высказываний – любое непустое множество, элементы которого есть символы трех категорий:

- символы первой категории: $x, y, z, \ldots, x_1, x_2, \ldots$ Эти символы будем называть *переменными высказываниями.* – буквы латинского алфавита с индексом или без него.
- символы второй категории: $\vee, \wedge, \to, \neg$. они носят общее название **логических связок.** Первый из них – знак дизъюнкции или логического сложения, второй – знак

конъюнкции или логического умножения, третий – знак импликации или логического следования и четвертый – знак отрицания.

- третью категорию составляет пара символов ( ), называемая скобками или разделитель.

Выполнимая формула – формула, которая принимает значение истина (И), на некотором наборе распределения истинностных значений.

Высказывание - это повествовательное предложение, о котором можно сказать, что оно истинно или ложно.

Двойственные формулы – формулы вида:
формула A^* получается из формулы А путем замены в ней каждой операции на двойственную.

Дизъюнкция двух высказываний А и В - высказывание, ложное тогда и только тогда, когда оба высказывания ложны.

Импликация двух высказываний А и В - высказывание, ложное тогда и только тогда, когда А - истинно, а В - ложно.

Интерпретация формулы – конкретный набор истинностных значений, приписанных переменных

Исчисление высказываний – это аксиоматическая логическая система, интерпретацией которой является алгебра высказываний.

Конъюнкция двух высказываний А и В - высказывание, истинное тогда и только тогда, когда истинны оба высказывания.

Логика высказываний - раздел логики, в котором вопрос об истинности или ложности высказываний рассматривается и решается на основе изучения способа построения высказываний из элементарных высказываний с помощью логических операций конъюнкции ("и"), дизъюнкции ("или"), отрицания ("не"), импликации ("если..., то...") и др.

Логически эквивалентны (равносильные) формулы – две формулы A и B, которые принимают одинаковые логические значения при любом наборе значений входящих в формулы элементарных высказываний или тогда и только тогда, когда формула $A \leftrightarrow B$ – тавтология.

Опровержимая формула – формула, которая принимает значение ложь (Л), на некотором распределении истинностных значений.

Отрицание высказывания А - высказывание, истинное тогда и только тогда, когда высказывание А ложно.

Приоритет связок – порядок выполнения логических операций, если скобки не указаны, устанавливаемый в виде:

$\neg, \quad \wedge, \vee, \rightarrow, \leftrightarrow$.

Подформулой элементарной формулы является сама формула,

- если формула имеет вид $\overline{A}$, то ее подформулами являются: она сама, формула А и все подформулы формулы А.
- если формула имеет вид (А*В) (здесь и в дальнейшем под символом * будем понимать любой из трех символов $\vee,\wedge,\rightarrow$), то ее подформулами являются: она сама, формулы А и В и все подформулы формул А и В.

Противоречие – формула, ложная при любых значениях букв.

Пропозициональные связки - символы отрицание, конъюнкция, дизъюнкция, импликация и эквивалентность.

Составные высказывания или сложные высказывания – высказывания, которые можно выразить из элементарных высказываний с помощью связок «не», «и», или», «если то», «тогда и только тогда».

Тавтология – формула, которая принимает только истинные значения при любых значениях букв.

Формула – правильно построеное составное высказывание:

1) Всякая буква есть **формула**.

3) Если A, B - формулы, то формулами являются также $\neg A$, $A \vee B$, $A \wedge B$, $A \rightarrow B$, $A \leftrightarrow B$.

Элементарные высказывания или простые высказывания – высказывания, представляющие одно утверждение, не могут быть выражены через другие высказывания.

Эквивалентность(эквиваленция) двух высказываний А и В - высказывание, истинное тогда и только тогда, когда истинностные значения А и В совпадают.

Язык исчисления высказываний – это исходные символы, из которых строятся формулы исчисления высказываний.

Аксиоматический метод - способ построения научной теории, при котором в её основу кладутся некоторые исходные положения (суждения) — аксиомы, из которых все остальные утверждения этой науки должны выводиться чисто логическим путём, посредством доказательств.

Вывод формальной теории - последовательность формул A_1, A_2, …, A_n, в которой все формулы – либо аксиомы, либо получаются из предыдущих по правилам вывода.

Выводимая формула A из множества формул Γ ($\Gamma \vdash A$) - существует вывод A_1, A_2, …, A_n, где $A_n = A$, и есть три возможности: $A_i \in \Gamma$; A_i - аксиома; A_i получаются из предыдущих формул по правилам вывода.

Интерпретация теории - приписывание значений первичным понятиям аксиоматической теории.

Непротиворечивое исчисление высказываний - исчисление высказываний, в котором существует такая формула *A*, что не доказуема и сама формула и ее отрицание.

Неформальные или интуитивные теории - аксиоматические теории, в которых правила логики явно не заданы.

Общезначимая формула – формула, истинная в любой интерпретации.

Вывод формул - процесс получения доказуемых формул.

Выводимая (доказуемая) формула:

а) Всякая аксиома является доказуемой формулой.

б)Формула, полученная из доказуемой формулы путем применения подстановки вместо переменной x произвольной формулы B, есть доказуемая формула.

в) Формула B, полученная из доказуемых формул A и $A \to B$ путем применения ПЗ, есть доказуемая формула.

г) Никакая другая формула исчисления высказываний не считается доказуемой .

Правило заключения (ПЗ):Если формулы А и А→В выводимы (доказуемы) в исчислении высказываний, то формула В также выводима (доказуема).

$$\frac{\mathbf{A};\ \vdash \mathbf{A}\to\mathbf{B}}{\vdash \mathbf{B}} \qquad \textbf{(Modus ponens)}$$

«если верно, что из A следует B и A является истинным, то истинно B»

Правило контр позиции: Если доказуема формула А→В, то доказуема формула $\overline{B} \to \overline{A}$, т. е.

$$\frac{\vdash \mathbf{A}\to\mathbf{B}}{\vdash \overline{B}\to\overline{A}}$$

Правило подстановки(ПП): Если формула А выводима (доказуема) в исчислении высказываний, х- переменная, В-произвольная формула исчисления высказываний, то формула , полученная в результате замены в формуле А переменной х всюду, где она входит, формулой В, является также выводимой(доказуемой) формулой.

$$\int_{X}^{B}(A)\ .$$

Правило силлогизма: Если доказуемы формулы А→В и В→С, то доказуема формула А→С , т. е.

$$\frac{\vdash \mathbf{A}\to\mathbf{B},\ \vdash \mathbf{B}\to\mathbf{C}}{\vdash \mathbf{A}\to\mathbf{C}}$$

Правило сложного заключения: Если формулы $A_1, A_2, ..., A_n$ и $A_1 \to (A_2 \to (A_3 \to (..(A_n \to L)...)))$ доказуемы, то и формула L доказуема.

$$\frac{\vdash \mathbf{A_1},\ \vdash \mathbf{A_2},\ \ldots,\ \vdash \mathbf{A_n},\ \vdash \mathbf{A_1}\to(\mathbf{A_2}\to(\mathbf{A_3}\to(\ldots(\mathbf{A_n}\to\mathbf{L})\ \ldots)))}{\vdash \mathbf{L}}$$

Правило сложной подстановки (СПП): Пусть А – доказуемая формула; $x_1, \ldots, x_n$ - переменные, а $B_1, \ldots, B_n$ - любые формулы ИВ. Тогда результат одновременной подстановки в формулу А вместо $x_1, \ldots, x_n$ соответственно формул $B_1, \ldots, B_n$ является доказуемой формулой.

$$\vdash \mathbf{A}$$

$$\vdash \int_{X_1,\ldots X_n}^{B_1,\ldots,B_n} (A)$$

Правило снятия двойного отрицания: а) Если доказуема формула $A \to \overline{\overline{B}}$, то доказуема формула $A \to B$.

б) Если доказуема формула $\overline{\overline{A}} \to B$, то доказуема формула $A \to B$.

$$\frac{\vdash A \to \overline{\overline{B}}}{\vdash A \to B} \qquad \text{и} \qquad \frac{\vdash \overline{\overline{A}} \to B}{\vdash A \to B}$$

Система аксиом исчисления высказываний состоит из 11 аксиом:

Первая группа аксиом (содержащая только импликацию).

I_1: $x \to (y \to x)$.

I_2: $(x \to (y \to z)) \to ((x \to y) \to (x \to z))$.

Вторая группа аксиом (к импликации присоединилась конъюнкция):

II_1: $x \wedge y \to x$.

II_2: $x \wedge y \to y$.

II_3: $(z \to x) \to ((z \to y) \to (z \to x \wedge y))$.

Третья группа аксиом (к импликации присоединилась дизъюнкция):

III_1: $x \to x \vee y$

III_2: $y \to x \vee y$.

III_3: $(x \to z) \to ((y \to z) \to (x \vee y \to z))$.

Четвертая группа аксиом (к импликации присоединилось отрицание):

IV_1: $(x \to y) \to (\bar{y} \to \bar{x})$.

IV_2: $x \to \overline{\overline{x}}$

IV_3: $\overline{\overline{x}} \to x$.

Формула, выводимая из совокупности гипотез Н

1)Всякая формула $A_i \in H$,является формулой, выводимой из Н.
2) Всякая доказуемая формула выводима из Н.
3) Если формулы С и С→В выводимы из совокупности Н, то формула В также выводима из Н.

Вывод из конечной совокупности формул Н - всякая конечная последовательность формул $B_1,B_2,\ldots,B_к$, всякий член которой удовлетворяет одному из следующих трех условий:

1. является одной из формул совокупности Н,
2. является доказуемой формулой,
3. получается по правилу заключения из двух любых предшествующих членов последовательности В1,В2,…,Вк

Основные правила выводимости:

1. $H \vdash A$
2. $H,C \vdash A, H \vdash C$ / $H \vdash A$.
3. $H,C \vdash A,\ W \vdash C$ / $H,W \vdash A$
4. $H \vdash C \to A$ / $H,C \vdash A$

5. Теорема дедукции: Пусть H - множество формул, C, A – формулы, тогда $H, C \vdash A$. / $H \vdash C \to A$

В частности, если $H = \varnothing$, то если $C \vdash A \Rightarrow C \to A$

5А. Обобщенная теорема дедукции:

$\{C1, C1, \ldots, Ck\} \vdash A$ / $\vdash C1 \to (C2 \to (C3 \to \ldots (Ck \to A)\ldots))$

Теорема. (обратная теорема дедукции)

$$H \vdash A \to B \Rightarrow H,\ A \vdash B.$$

6. Правило введения конъюнкции: $H \vdash A, H \vdash B$ / $H \vdash A \wedge B$

7. Правило введения дизъюнкции:

$H,A \vdash C; H,B \vdash C$. / $H, A \vee B \vdash C$

Аксиоматическое исчисление высказываний называется полным в узком смысле, если добавление к списку его аксиом любой недоказуемой в исчислении формулы в качестве новой аксиомы приводит к противоречивому исчислению.

Исчисление высказываний называется полным в широком смысле, если любая тождественно истинная формула в нем доказуема.

Независимая аксиома – аксиома, которая не может быть выведена из остальных аксиом

Независимая система аксиом исчисления – каждая аксиома системы независима

Непротиворечивое логическое исчисление – логическое исчисление, в котором не доказуемы никакие две формулы, из которых одна является отрицанием другой

Автоматическое доказательство формулы - алгоритм, который проверяет вывод

$$A_1, A_2, \ldots, A_n \vdash B.$$

Если посылки $A_1, A_2, \ldots, A_n$ истинны, то истинно заключение B.

Дизъюнкт – это дизъюнкция литер (или элементарная дизъюнкция).

Контрарные литеры -литеры A и $\neg A$, а множество $\{A, \neg A\}$ – *контрарная пара.*

Литерой будем называть выражения A или $\neg A$.

Метод резолюций - аксиоматическая теория первого порядка, которая использует доказательство от противного, и, не использует аксиоматику исчисления предикатов.

Правило резолюций - правило вывода:

$$\frac{A \vee B,\ \neg A \vee C}{B \vee C}.$$

Предложение - дизъюнкция формул вида A или $\neg A$.

Пустой дизъюнкт (обозначается) – дизъюнкт, не содержащий литер.

Резольвента дизъюнктов $A \vee B$ и $\neg A \vee C$ по литере A - дизъюнкт $B \vee C$: $B \vee C = res_A(A \vee B \text{ и } \neg A \vee C)$.

Двухместный предикат P(x,y) - функция двух переменных x и y, определенная на множестве $M=M_1 x M_2$ и принимающая значения из множества {1;0}.

Дизъюнкция двух предикатов P(x) и Q(x) - предикат $P(x) \vee Q(x)$, который принимает значение "ложь" при тех и только тех значениях $x \in M$, при которых каждый из предикатов принимает значение "ложь", и принимает значение "истина" во всех остальных случаях.

Импликация предикатов P(x) и Q(x) - предикат $P(x) \to Q(x)$, который является ложным при тех и только тех значениях $x \in M$, при которых одновременно P(x) принимает значение "истина", а Q(x) – значение "ложь", и принимает значение "истина" во всех остальных случаях.

Конъюнкция двух предикатов P(x) и Q(x) - предикат $P(x) \wedge Q(x)$, который принимает значение "истина" при тех и только тех значениях $x \in M$, при которых каждый из предикатов принимает значение "истина", и принимает значение "ложь" во всех остальных случаях.

Множество (область) истинности предиката P(x) - множество всех элементов $x \in M$, при которых предикат принимает значения "истина" (1), т.е. множество истинности предиката P(x)- это множество

$$I_p = \{x : x \in M, P(x) = 1\},$$

n-местный предикат - это функция $P(x_1, x_2, \ldots x_n)$ определенная на наборах длины n элементов некоторого множества M, принимающая значения в области True, False. Множество M называется *предметной областью предиката*, а x1, x2, ..xn *–предметными переменными*.

$$P(x_1, x_2, \ldots x_n) \to \{и, л\}.$$

Область определения предиката - множество M, на котором определен предикат P(x).

Одноместный предикат P(x) - произвольная функция переменного x, определенная на множестве M и принимающая значение из множества {1; 0}.

Отрицание предиката P(x) - предикат $\overline{P}(x)$ или $\overline{P(x)}$, который принимает значение "истина" при всех значениях $x \in M$, при которых предикат P(x) принимает значение "ложь", и принимает значение "ложь" при тех значениях $x \in M$, при которых предикат P(x) принимает значение "истина".

Предикат - это высказывание,функция, значение (истина/ложь) которого зависит от параметров.

Тождественно истинный предикат – предикат, множество истинности которого совпадает с областью определения.

Тождественно ложный предикат – предикат, множество истинности которого является пустым множеством.

Эквиваленция предикатов P(x) и Q(x) - предикат $P(x) \leftrightarrow Q(x)$, который обращается в "истину" при всех тех и только тех $x \in M$, при которых P(x) и Q(x) обращаются оба в истинные или оба в ложные высказывания.

Навешивание квантора - присоединение квантора с переменной к предикатной формуле, переменная при этом называется *связанной* и вместо нее подставлять константы уже нельзя.

Логика предикатов - раздел логических теорий, в котором изучаются общезначимые связи между высказываниями о свойствах и отношениях предметов; в основе лежит формализованный язык, отображающий субъективно-предикатную структуру высказываний.

Равносильные формулы на области M - две формулы логики предикатов A и B, принимающие одинаковые логические значения при всех значениях входящих в них переменных, отнесенных к области M.

Формулы логики предикатов:

7. Каждое высказывание как переменное, так и постоянное, является формулой (элементарной).
8. Если $F(\cdot,\cdot, \ldots,\cdot)$ – n-местная предикатная переменная или постоянный предикат, а $x_1, x_2,\ldots, x_n$– предметные переменные или предметные постоянные (не обязательно все различные), то $F(x_1, x_2,\ldots, x_n)$ есть формула. Такая формула

называется **элементарной**, в ней предметные переменные являются свободными, не связанными кванторами.

9. Если А и В – формулы, причем, такие, что одна и та же предметная переменная не является в одной из них связанной, а в другой – свободной, то слова $A \vee B, A \wedge B, A \to B$ есть формулы. В этих формулах те переменные, которые в исходных формулах были свободны, являются свободными, а те, которые были связанными, являются связанными.

10. Если А – формула, то $\overline{A}$ – формула, и характер предметных переменных при переходе от формулы А к формуле $\overline{A}$ не меняется.

11. Если А(х) – формула, в которую предметная переменная х входит свободно, то слова $\forall x A(x)$ и $\exists x A(x)$ являются формулами, причем, предметная переменная входит в них связанно.

Всякое слово, отличное от тех, которые названы формулами в пунктах 1 – 5, не является формулой.

Взаимно обратные друг другу теоремы - Пара теорем, у которых условие одной является заключением второй, а условие второй является заключением первой

Взаимно противоположные теоремы - Пара теорем, у которых условие и заключение одной являются отрицанием соответственно условия и заключения другой

Выполнимая формула в области М - формула А логики предикатов, в которой существуют значения переменных входящих в эту формулу и отнесенных к области М (иначе – существует модель), при которых формула А принимает истинные значения.

Общезначимая формула – формула А логики предикатов, которая тождественно истинна на всякой области (на любой модели)

Приведенная нормальная форма - формула логики предикатов, которая содержит только операции конъюнкции, дизъюнкции и кванторные операции, а операция отрицания отнесена к элементарным формулам.

Тождественно ложная формула - формула А логики предикатов, принимающая ложные значения для всех значений переменных, входящих в эту формулу и отнесенных к этой области

Степень принадлежности определяется так называемой функцией принадлежности **M(d),** где d-расстояние до помехи.

$A=\{\mu_A(x)/x\}$,

где $\mu_A(x)$ - *характеристическая функция принадлежности* (или просто функция принадлежности), принимающая значения в некотором вполне упорядоченном множестве М.

Фаззификация - сопоставление множества значений х ее функции принадлежности М(х), т.е. перевод значений х в нечеткий формат (пример с термином молодой).

Алгоритм - предписание, однозначно задающее процесс преобразования исходной информации в виде последовательности элементарных дискретных шагов, приводящих за их конечное число к результату.

Всюду определенная функция –функция, область определения которой из X в Y совпадает с множеством X.

Примитивно рекурсивная функция - частичная функция $f(x_1,\ldots, x_n)$, которую ее можно получить конечным числом операций суперпозиции и примитивной рекурсии, исходя лишь из простейших функций S^1, 0^n и I_m^n

Частично рекурсивная функция - частичная функция $f(x_1,\ldots, x_n)$, которую можно получить конечным числом операций суперпозиции, примитивной рекурсии и минимизации, исходя лишь из простейших функций S^1, 0^n и I_m^n.

Частичная функция из X в Y - некоторым элементам множества X поставлены в соответствие однозначно определенные элементы множества Y

Эффективно вычислимая функция – это такая функция $y(x_1, x_2, \ldots, x_n)$, для которой существует алгоритм, позволяющий вычислить ее значение по известным значениям аргументов

Алфавит - конечное, непустое множество элементов называемых буквами. Различные сочетания букв образуют слова.

Длина слова – число символов в слове

Нормальная схема подстановок - это конечный набор, состоящий из пар слов, где левое слово переходит в правое (но не наоборот)

Подслово - слово s называется ***подсловом*** слова q, если q можно представить в виде

q=rst,

где **r и t** – любые слова в том же алфавите (в том числе и пустые)

Подстановкой называется замена первого по порядку подслова P_r исходного слова P на слово P_k

Пустое слово – слово, длина которого равна нулю

Слово - это любая конечная последовательность знаков алфавита.

Конфигурация машины - совокупность состояний всех ячеек ленты, состояния ЛУ и положение головки.

Полиномиальный алгоритм (или алгоритм полиномиальной временной сложности, или алгоритм принадлежащим классу **P**) - алгоритм, у которого временная сложность равна $O(n^k)$, где k - целое число > 0

Экспоненциальные алгоритмы – алгоритмы, , для временной сложности которых не существует такой оценки

Труднорешаемая задача – задача, для которой не существует разрешающего полиномиального алгоритма.

Список литературы

Основная литература

1. Гринченков, Д.В. Математическая логика и теория алгоритмов для программистов: Учебное пособие / Д.В. Гринченков, С.И. Потоцкий. - М.: КноРус, 2013. - 206 с.
2. Идиатулин, В.С. Математическая логика. Курс лекций. Задачник-практикум и решения: Учебное пособие. 4-е изд. / В.С. Идиатулин. - СПб.: Лань П, 2016. - 288 с.
3. Колмогоров, А.Н. Математическая логика: Дополнительные главы / А.Н. Колмогоров, А.Г. Драгалин. - М.: УРСС, 2015. - 240 с.

Дополнительная литература

1. Ершов, Ю.Л. Математическая логика / Ю.Л. Ершов, Е.А. Палютин. - М.: Физматлит, 2011. - 356 с.
2. Глухов, М.М. Математическая логика. Дискретные функции. Теория алгоритмов. Учебное пособие / М.М. Глухов, А.Б. Шишков. - СПб.: Лань, 2012. - 416 с.
3. Гудстейн, Р.Л. Математическая логика / Р.Л. Гудстейн . - М.: ЛИБРОКОМ, 2010. - 160 с.

УДК 519.6

Рекомендовано к изданию методическим советом ПГУТИ, протокол №45, от 10.03.2017 г.

Блатов, И.А., Старожилова О.В.

Б Математическая логика и теория алгоритмов: учебное пособие / И.А.Блатов, О.В.Старожилова – Самара: ПГУТИ, 2017. –214 с.

Учебное пособие затрагивает такие разделы математической логики и теории алгоритмов как: алгебра высказываний, исчисление высказываний, логика предикатов, исчисление предикатов, элементы теории алгоритмов. Предназначено в качестве учебного пособия для студентов направления подготовки 09.03.02. «Информационные системы и технологии», а также для студентов и магистрантов других направлений подготовки и специалистов, желающих изучать математическую логику самостоятельно.

Каждый раздел заканчивается контрольными вопросами, которые помогут проверить теоретическое освоение курса, содержит большое количество задач для самостоятельного решения и ответы для проверки.